基于空间填充曲线网格划分的最近邻查询算法

徐红波[1] 郝忠孝[1,2]
(哈尔滨理工大学计算机科学与技术学院 哈尔滨 150080)[1]
(哈尔滨工业大学计算机科学与技术学院 哈尔滨 150001)[2]

摘 要 在建树过程中,R树存在最小边界矩形之间重叠的现象。当数据量较大时,重叠现象尤为严重,基于R树最近邻查询算法的性能急剧恶化。针对该问题,利用空间填充曲线的降低维度特性和数据聚类特性,提出一种基于网格划分最近邻查询算法。该算法将整个数据空间划分成大小相等、互不重叠的网格,对网格中的点进行线性排序之后,只需要访问查询点所在网格中的点及其周边邻近网格中的点,就能够获得最近邻。在Hilbert曲线、Z曲线和Gray曲线上实现3种最近邻查询算法,在映射算法和数据聚类特性上实验比较3种曲线之间的性能差异。实验结果表明,算法的查询性能明显优于顺序扫描算法和基于R树的最近邻查询算法。

关键词 空间填充曲线,网格划分,最近邻,降维
中图法分类号 TP311.13 **文献标识码** A

Nearest-neighbor Query Algorithm Based on Grid Partition of Space-filling Curve

XU Hong-bo[1] HAO Zhong-xiao[1,2]
(College of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China)[1]
(College of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)[2]

Abstract As the overlap between minimum bounding rectangles in the directory of R-tree is increasing very rapidly with growing number of the data, the performance of the nearest-neighbor query algorithm based on R-tree deteriorates rapidly. To avoid the problem, the paper presented a nearest-neighbor query algorithm based on grid partition of space-filling curve. Space-filling curve has the properties of dimension reduction and data clustering. Using space-filling curve, the algorithm divides 2D space into equal-size grids, and map points in grids into linear space. It only visits points in the grid which query point lies in and points in near grids of query point. The paper implemented nearest-neighbor query algorithms based on Hilbert curve, Z curve and Gray curve, and compared the difference of mapping algorithm and data clustering between three curves. Experimental results indicate that the algorithm is better than brute-force algorithm and near-neighbor query algorithm based on R-tree.

Keywords Nearest neighbors, Grid, Space-filling curve, Reduction of dimensionality

最近邻查询是地理信息系统、数据挖掘和模式识别等领域中重要的问题之一。

最简单的最近邻查询算法为顺序扫描(Brute-Force)算法,该算法依次计算查询点与点集中其他点的距离,从中选出与查询点距离最近的k个点,也就是k最近邻。

文献[1]提出分枝限界k最近邻查询算法。该算法深度优先(Depth-First)遍历R树,在递归和回溯过程中利用基于距离函数mindist()和minmaxdist()的3条准则将结点中不包含k最近邻的子树剪掉,从而避免了对整棵R树的遍历。文献[2]在文献[1]的基础上提出最短优先(Best-First)最近邻查询算法。该算法优先遍历距离查询点最近的结点,从而结点访问数明显少于深度优先算法。文献[3]证明3条准则中的两条不能够高效地删除结点的子树,提出依据一条准则的最近邻查询算法。文献[4]在文献[2,3]的基础上提出多个对象最近邻查询算法。多对象最近邻查询是根据某几个点,找出离它们最近的一个点或者k个点。该算法将多对象k最近邻转化成这些对象几何中心点的k最近邻。

文献[5]提出基于Hilbert曲线的近似k最近邻查询算法,并分析了近似k最近邻的误差。

在建树过程中,R树存在最小边界矩形之间重叠的现象。当数据量较大时,重叠现象尤为严重,基于R树最近邻查询算法的性能急剧恶化。针对该问题,本文利用空间填充曲线的降低维度特性和数据聚类特性,将整个数据空间划分成大小相等、互不重叠的网格,并对网格中的点进行线性排序。最近邻查询算法只需访问查询点所在网格中的点及其周边邻近网格中的点,就能获得查询点的最近邻。

1 基础知识

多维空间索引结构的设计一般来说难于线性空间索引结构的设计,原因在于在多维空间中无法用线性顺序来保持空

到稿日期:2009-02-09 返修日期:2009-05-01 本文受黑龙江省自然科学基金(F200601)资助。
徐红波(1980—),男,博士研究生,讲师,主要研究方向为空间数据库索引结构及查询算法,E-mail:x_h_b@tom.com;郝忠孝(1940—),男,教授,博士生导师,主要研究方向为数据库理论及应用。

间对象之间的位置关系。二维空间中对象之间的位置关系大致为4种情况，即东西南北，而三维空间中对象之间的位置关系就更多了。在多维空间中折中的方法是寻找一种映射关系，它至少能够在一定程度上保持空间对象之间的位置关系。在这节中使用空间填充曲线将空间对象映射到线性空间中。

空间填充曲线是一种降低空间维度的方法。它像一条线一样穿过空间中每个离散网格，按照线性顺序对这些网格进行编号。空间填充曲线主要有 Hilbert 曲线、Z 曲线和 Gray 曲线。Hilbert 曲线的数据聚类特性最优，Z 曲线的数据聚类特性最差；Hilbert 曲线的映射过程最复杂，Z 曲线的映射过程最简单。3种曲线的映射算法详见文献[6]。

基本空间填充曲线沿着一条路径填充边长为2的 d 维超矩形。这条路径只访问网格一次，且路径互不交叉。路径存在一个开始点和一个结束点，这两个端点可以和其它路径的端点相连接，形成规模更大的曲线。基本曲线称为1阶曲线。为了得到 m 阶曲线，基本曲线的每个网格被 $m-1$ 阶曲线填充。这种填充过程需要曲线进行必要反射和旋转操作来适应新曲线的路径。

基本 Z 曲线如图1(a)所示。为了获得 m 阶 Z 曲线，则将基本 Z 曲线中网格 G_0，G_1，G_2 和 G_3 由 $m-1$ 阶 Z 曲线进行填充。1阶 Z 曲线中的4个网格由1阶 Z 曲线进行填充得到2阶 Z 曲线，2阶 Z 曲线如图1(b)所示；1阶 Z 曲线中的4个网格由2阶 Z 曲线进行填充得到3阶 Z 曲线，3阶 Z 曲线如图1(c)所示。Z 曲线的填充过程无需任何反射和旋转操作。

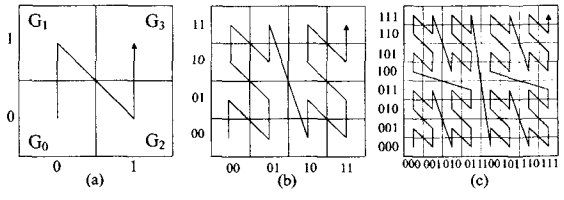

图1 Z 曲线

1阶 Gray 曲线如图2(a)所示。为了获得 m 阶 Gray 曲线，将1阶 Gray 曲线中的网格由 $m-1$ 阶 Gray 曲线进行填充，同时 $m-1$ 阶 Gray 曲线必须进行旋转操作。

Gray 曲线的旋转规则：在 G_0 和 G_2 中 $m-1$ 阶曲线无变化；在 G_1 和 G_3 中 $m-1$ 阶曲线逆时针旋转180°。2阶和3阶 Gray 曲线如图2(b)和图2(c)所示。

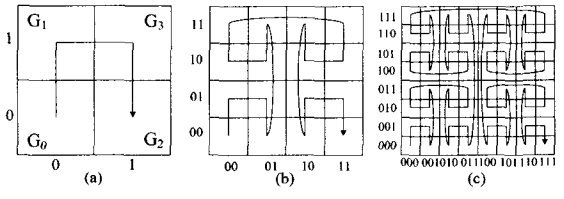

图2 Gray 曲线

1阶 Hilbert 曲线如图3(a)所示，为了获得 m 阶 Hilbert 曲线，将基本 Hilbert 曲线中的每个网格由 $m-1$ 阶 Hilbert 曲线进行填充，同时 $m-1$ 阶 Hilbert 曲线必须进行相应的反射和旋转操作。

Hilbert 曲线的反射和旋转规则：在 G_0 中 $m-1$ 阶曲线先反射，然后顺时针旋转90°；在 G_1 和 G_3 中 $m-1$ 阶曲线无变化；在 G_2 中 $m-1$ 阶曲线先反射，然后逆时针旋转90°。2阶和3阶 Hilbert 曲线如图3(b)和图3(c)所示。

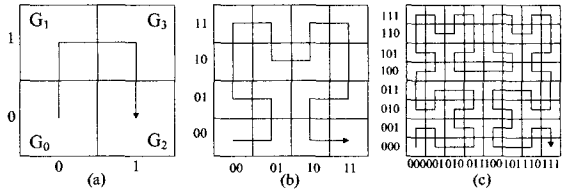

图3 Hilbert 曲线

当空间维度 d 为定值时曲线的阶 m 决定了空间分割的粒度，m 越大，网格越小。m 阶曲线将二维空间分割成大小相等的 2^{2m} 个网格。同理，m 阶曲线将 d 维空间分割成大小相等的 2^{dm} 个网格。

平面中点有两个坐标：x 坐标和 y 坐标。空间填充曲线对空间的分割，等同于在 x 坐标值和 y 坐标值之间交替分割。1阶曲线将平面分割成4个面积相等的网格，网格的 x 坐标值 x_0 和 y 坐标值 y_0 从0和1取值。

坐标值的二进制形式中每一位表示在 x 轴或 y 轴上一次分割。m 阶空间填充曲线相当于在坐标轴上分割 m 次，此时用位串表示网格，位串的形式为 $(x_{m-1}x_{m-2}\cdots x_1x_0, y_{m-1}y_{m-2}\cdots y_1y_0)_2$。

在 Z 曲线中网格的坐标值通过位交叉操作形成网格的 Z 值。Z 值描述了网格的位序。

设网格的 x 坐标 $x=(1)_{10}=(01)_2=(x_1x_0)_2$ 和 y 坐标 $y=(3)_{10}=(11)_2=(y_1y_0)_2$，$()_2$ 表示数的二进制形式，$()_{10}$ 表示数的十进制形式。通过位交叉操作，Z 值 $(x_1y_1x_0y_0)_2=(0111)_2$。$G$ 值、H 值用于 Gray 曲线和 Hilbert 曲线[6]。

2阶 Z 曲线的位交叉操作如图4所示。

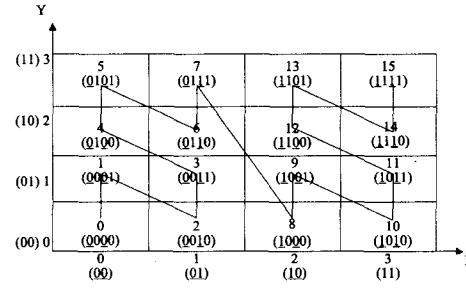

图4 2阶 Z 曲线

2 最近邻查询算法

最近邻查询算法的思想是，只访问查询点所在的网格及其周边网格中的点，从而找到查询点的最近邻点。

例1 最近邻查询算法的示例如图5所示。在 Z 曲线中查询点 q 所在网格的 Z 值为24，邻近网格的集合为{7,13,15,18,19,25,26,27}，存在5个聚集{7},{13},{15},{18,19}和{24,25,26,27}；在 Gray 曲线中查询点 q 所在网格的 G 值为20，邻近网格的集合为{4,8,11,21,22,23,26,27}，存在5个聚集{4},{8},{11},{20,21,22,23}和{26,27}；在 Hilbert 曲线中查询点 q 所在网格的 H 值为30，邻近网格的集合为{10,11,12,17,18,28,29,31}，存在3个聚集{10,11,12},{17,18},{28,29,30,31}。

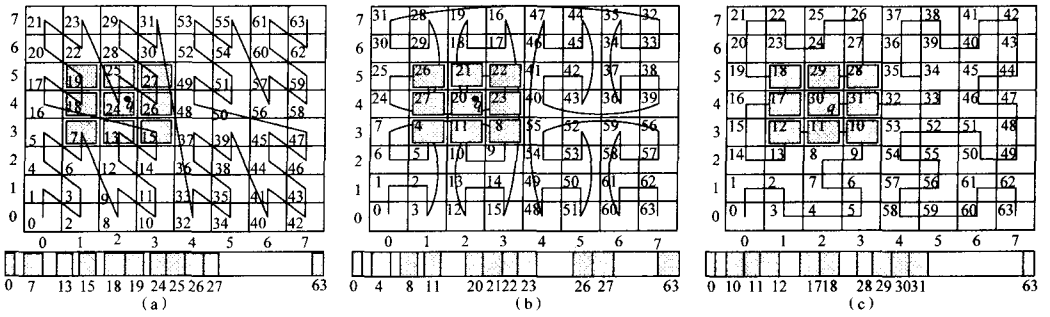

图 5　空间填充曲线最近邻查询

最近邻查询算法的实现难点在于如何求得 Gray 曲线和 Hilbert 曲线中邻近网格对应的 G 值和 H 值。

本文提出两个转换算法，GNN 和 HNN，依据 Z 曲线与 Gray 曲线和 Hilbert 曲线之间的关系快速求得 G 值和 H 值。

表 1 给出方向与坐标之间的关系。

表 1　方向与坐标之间的关系

d \ c	n	e	s	w	en	es	ws	wn
x	x	x+1	x	x−1	x+1	x+1	x−1	x−1
y	y+1	y	y−1	y	y+1	y−1	y−1	y+1

基于 Z 曲线最近邻查询算法：第一步，对 Z 值进行逆交叉操作，得到对应的网格坐标；第二步，按照方向参数求得近邻网格的坐标；第三步，对网格坐标进行位交叉操作，得到对应的 Z 值。

Algorithm ZNN(Z_{z2},d)

{

将 Z 值 Z_{z2} 进行逆位交叉操作得网格坐标 $(X,Y)_{c2}$；

将二进制 $(X,Y)_{c2}$ 转换成等值十进制 $(X,Y)_{c10}$；

由表 1 得网格 $(X,Y)_{c10}$ 在方向 d 上的网格 $(NX,NY)_{c10}$；

将十进制 $(NX,NY)_{c10}$ 转换成等值二进制 $(NX,NY)_{c2}$；

将网格 $(NX,NY)_{c2}$ 进行位交叉操作得 Z 值 NZ_{z2}；

}

例 2　查询点 q 落在网格 $(2,4)_{10}$ 中，经过位交叉操作之后得到网格对应的 Z 值 $(24)_{z10}$。算法 ZNN 的具体执行过程如图 6 所示。

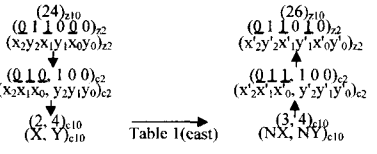

图 6　算法 ZNN($(011000)_{z2}$,"east")的执行过程

基于 Gray 曲线最近邻查询算法：第一步，将 G 值转换为对应的 Z 值；第二步，调用算法 ZNN 求得近邻网格对应的 Z 值；第三步，将 Z 值转换为对应的 G 值。

Algorithm GNN(G_{g2},d)

{　//曲线的阶为 m

//位串中从左到右的索引为从 1 到 $2m$

$Z_{z2}[1]=G_{g2}[1]$；

for i=2 to $2m$

$Z_{z2}[i]=G_{g2}[i]\oplus Z_{z2}[i-1]$;

NZ_{z2}=ZNN(Z_{z2},d)；

$NG_{g2}[1]=NZ_{z2}[1]$;

for i=2 to $2m$ do

$NG_{g2}[i]=NZ_{z2}[i]\oplus NZ_{z2}[i-1]$;

}

例 3　查询点 q 落在网格 $(2,4)_{10}$ 中，经过映射算法之后得到网格对应的 G 值 $(20)_{g10}$。算法 GNN 的具体执行过程如图 7 所示。

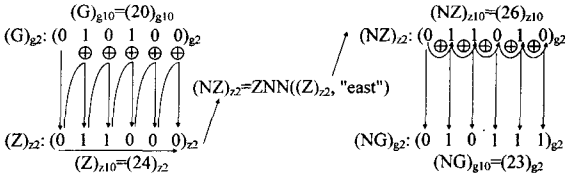

图 7　算法 GNN($(010100)_{g2}$,"east")的执行过程

基于 Hilbert 曲线最近邻查询算法：第一步，将 H 值转换为对应的 Z 值；第二步，调用算法 ZNN 求得近邻网格对应的 Z 值；第三步，将 Z 值转换为对应的 H 值。

Algorithm HNN(H_{h2},d)

{

长度为 m 的数组 $A1$ 和 $A2$；

从左到右将 H_{h2} 分割成 m 个长度为 2 的子串；

依次存放到数组 $A1$ 和 $A2$ 中；

调用函数 RRT($A1$,$A2$)将曲线从 m 阶转换为 1 阶；

调用函数 ZHT($A2$)将 H 值转换为 Z 值；

将数组 $A2$ 中的每个元素链接成位串 Z_{z2}；

NZ_{z2}=ZNN(Z_{z2},d)；

从左到右将 NZ_{Z2} 分割成 m 个长度为 2 的子串；

依次存放到数组 $A1$ 中；

调用函数 ZHT($A1$)将 Z 值转换为 H 值；

调用函数 RRT($A1$,$A2$)将 1 阶曲线转换为 m 阶；

将数组 $A2$ 中每个元素链接成位串 NH_{h2}；

}

例 4　查询点 q 落在网格 $(2,4)_{10}$ 中，经过映射算法之后得到网格对应的 H 值 $(30)_{h10}$。算法 HNN 的具体执行过程如图 8 所示。

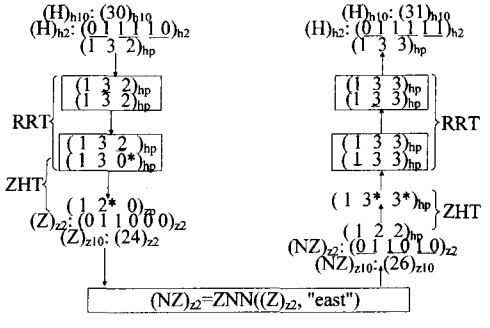

图 8　算法 HNN($(011110)_{h2}$,"east")的执行过程

Hilbert 曲线的反射和旋转操作如图 9 所示。用 1 阶曲线填充 1 阶曲线的左下角网格时，首先将曲线向右反射，然后顺时针旋转 90°，具体的过程如图 9(b)所示；用 1 阶曲线填充 1 阶曲线的右下角网格时，首先将曲线向左反射，然后逆时针旋转 90°，具体的过程如图 9(c)所示。

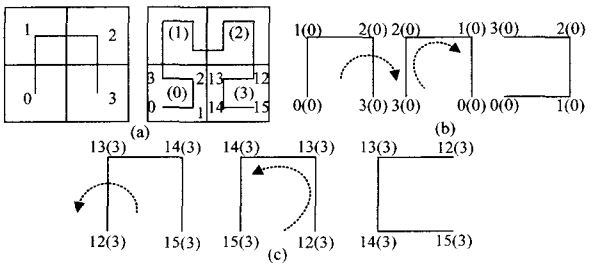

图 9　Hilbert 曲线的反射和旋转操作

算法 RRT 将 Hilbert 曲线从 m 阶转换为 1 阶。

```
Algorithm RRT(in,out)
{
  for i=1 to in.length-1
    if(in[i]==0) then
      for j=i+1 to out.length
        if(out[j]==1) then out[j]=3
          else if(out[j]==3) then out[j]=1
    else if(in[i]==3) then
          for j=i+1 to out.length
            if(out[j]==0) then out[j]=2
            else if(out[j]==2) then out[j]=0;
}
```

1 阶 Hilbert 曲线与 Z 曲线之间的关系如图 10 所示。

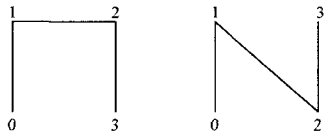

图 10　Hilbert 曲线与 Z 曲线之间的关系

Hilbert 曲线与 Z 曲线之间的转换操作应该将 2 和 3 互换。

算法 ZHT 实现了 Z 曲线与 Hilbert 曲线之间的转换。

```
Algorithm ZHT(arr)
{
  for i=1 to arr.length do
    if(arr[i]==2) then arr[i]=3
    else if(arr[i]==3) then arr[i]=2;
}
```

定理 1　最近邻查询算法的时间复杂度为 $O(9(n/2^{2m})$，空间复杂度为 $O(n)$。

证明：假设点的分布是均匀的，曲线的阶为 m，点数为 n。算法只需访问查询点所在的网格及其四周网格中的点，在 2 维空间中只需访问 9 个网格中的点。m 阶曲线将 2 维空间分割成 2^{2m} 个网格，平均每个网格中存在 $n/2^{2m}$ 个点，故算法的时间复杂度为 $O(9(n/2^{2m})$。数据结构只需存储点的 id 和点对应的映射值(Z 值、G 值和 H 值)，故算法的空间复杂度为 $O(n)$。

3　实验结果

实验环境：1.86GHz 赛扬 540 处理器、1G 内存、120G 硬盘、Windows XP 操作系统、Visual C++. net 2003 编程工具和 Microsoft Access 2002 关系数据库管理系统。

测试数据：实际数据为旧金山道路网络结点数据[1)]，其数据量为 174955；人工合成数据由随机函数产生，满足均匀分布点数据，其数据量从 10 万到 100 万不等，如图 11 所示。

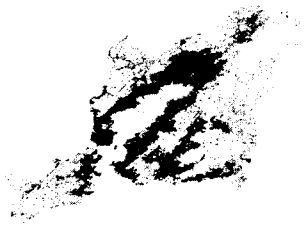

图 11　旧金山道路网络结点的分布

样本由随机函数从点集中随机选择的 1000 个点组成。

实验 1

算法效率的衡量指标包括 cpu 时间和 io 时间。在最近邻查询算法中，算法 ZNN，GNN 和 HNN 决定了 cpu 时间，近邻网格的聚集数决定了 io 时间。

在算法 GNN 和 HNN 中调用了算法 ZNN，故它们的执行时间多于 ZNN。而 HNN 的过程最复杂，其时间也最长。

将样本中每个点作为查询点，计算 ZNN，GNN 和 HNN 的执行时间和对应的聚集数，将总的时间和聚集数除以点数得到平均数。随着曲线阶的不同，算法的执行时间和曲线的聚集数如表 2 和 3 所列。

表 2　算法 ZNN，GNN 和 HNN 的平均执行时间

t((s)	4	5	6	7	8
ZNN	6	6	6	6	6
GNN	6	6	6	6	6
HNN	9	9	10	11	11

$n=174955, d=2$

由表 2 知，Z 曲线的时间最短，Hilbert 曲线的时间最长。

表 3　曲线的平均聚集数

cluster	4	5	6	7	8
ZNN	4.553	4.523	4.505	4.505	4.515
GNN	4.016	3.911	3.948	3.953	3.946
HNN	3.009	3.026	2.993	3.005	3.016

$n=174955, d=2$

由表 3 知，Hilbert 曲线的聚集数最小，Z 曲线的聚集数最大。

实验 2

在执行时间上比较基于 3 种空间填充曲线的最近邻查询算法与线性扫描、基于 R 树最短优先最近邻查询算法之间的差异。随着最近邻数的不同，算法的执行时间如表 4 所列。随着点数的不同，算法的执行时间如表 5 所列。

表 4　参数 k 与执行时间的关系

t(s)	1	2	3	4	5	6
Brute-force	0.9359	0.9328	0.9453	0.9375	0.939	0.9407
R-tree	0.0219	0.025	0.0219	0.025	0.025	0.0265

1) http://cs-people.bu.edu/lifeifei/tpq.html

Z	0.0047	0.0047	0.0047	0.0047	0.0062	0.0062
Gray	0.0031	0.0047	0.0047	0.0063	0.0062	0.0078
Hilbert	0.0047	0.0047	0.0047	0.0062	0.0063	0.0078

$n=174955, d=2, M=50, m=8$

从表 4 可知，曲线的执行时间少于线性扫描和基于 R 树最短优先最近邻查询算法的执行时间。

表 5 参数 n 与执行时间之间的关系

t(s)	10 万	20 万	50 万	60 万	80 万	1 百万
Brute-force	0.539	1.073	2.7078	3.2344	4.264	5.3625
R-tree	0.0531	0.1109	0.2734	0.3219	0.4297	0.575
Z	0.0015	0.0015	0.0031	0.0031	0.0047	0.0062
Gray	0.0016	0.0015	0.0031	0.0031	0.0047	0.0078
Hilbert	0.0015	0.0015	0.0031	0.0031	0.0047	0.0047

$d=2, k=1, M=50, m=8$

从表 5 知，曲线的执行时间最短。随着参数 n 的增加，曲线的执行时间线性增长，而线性扫描和基于 R 树最短优先最近邻查询算法的执行时间成倍增加。

结束语 本文提出了基于空间填充曲线网格划分最近邻查询算法。实验结果表明，该算法的性能优于线性扫描和基于 R 树最近邻查询算法。且算法可行、有效。

参 考 文 献

[1] Roussopoulos N, Kelley S, Vincent F. Nearest Neighbor Queries [C]// Proceedings of the 1995th ACM SIGMOD International Conference on Management of Data. San Jose, CA, 1995: 71-79

[2] Hjaltason G, Samet H. Incremental Distance Join Algorithms for Spatial Databases[C]// Proceedings of the 1998th ACM SIGMOD International Conference on Management of Data. Seattle, Washington, 1998: 237-248

[3] Cheung King Lum, Fu Ada Wai-chee. Enhanced nearest neighbour search on the R-tree[J]. ACM SIGMOD Record, 1998, 27(3): 16-21

[4] 刘永山，薄树奎，张强，等. 多对象的最近邻查询[J]. 计算机工程，2004，30(11)：66-68

[5] 徐红波，郝忠孝. 基于 Hilbert 曲线的近似 k-最近邻查询算法[J]. 计算机工程，2008，34(12)：47-49

[6] 徐红波. 空间填充曲线映射算法研究[J]. 科技信息，2007，24(35)：88-89

(上接第 183 页)

同义词转换处理；

第三步 将视频片断标注信息读入到 videoClip 中；

第四步 比较keyWords. Objects 和 videoClip. Objects信息，根据式(3)计算对象相似度 O，如果 $O=0$，则转到第八步；

第五步 比较keyWords. Actions 和 videoClip. Actions 信息，根据式(2)计算动作相似度 A；

第六步 比较 keyWords. Place 和 videoClip. Place 信息，计算地点相似度 P；

第七步 根据 OAP 模型，利用式(1)计算整个相似度 *Match*；

第八步 若所有视频未匹配完毕，则指向下一个视频片断，转到第三步；

第九步 按照 *Match* 取值大小，进行排序；

第十步 选择 *Match* 的最大值作为候选视频，若 *Match* 的最大值出现相等的情况，则用随机概率模型选择其中之一作为候选视频；

第十一步 算法结束。

4 实验结果分析

新闻视频自动检索方法在战略对抗演习中进行了应用，演习中的实验环境如下。

新闻稿总数：54，56，57(演习中 3 个不同回合)；

标注字典库：789 个词条；

自动切分字典库(每方单独建立)：最长 557 条记录，最短 35 条记录，平均 128 条记录；

视频素材库大小：6130 个视频片断，总计 23.5G。

实验结果如表 4 所列。在实验过程中对未被选中的视频进行跟踪记录。由于采用了基于关键字的 TBVR 技术，因此在查全率方面没有任何问题。同时，由于设计了同义词表，解决了语义匹配而字符不匹配的问题，因此取得了较高的查准率，但仍然存在少数遗漏的现象。对于这种现象需要补充和更新同义词表。

表 4 实验结果

测试集	查准率	查全率
54 条(回合 1)	94.7%	100%
56 条(回合 2)	94.5%	100%
57 条(回合 3)	95.1%	100%

结束语 本文在分析虚拟新闻系统新闻视频自动化检索的特点和现有视频检索方法不足的基础上，提出了一种基于 TBVR 的新闻视频自动检索方法，并对一般的 TBVR 中字符不匹配而语义却匹配的问题设计了解决方案，同时设计了视频相似度计算模型。实验结果表明，该方法能够很好地解决虚拟新闻系统中视频自动检索的问题，同时对于其他类似的视频检索问题有一定的借鉴和参考意义。

参 考 文 献

[1] 唐波，刘雨，孙茂印. 基于数据库的视频检索[J]. 电视技术，2005(2)：20-24

[2] 肖平，黄薇，冯刚. 基于内容的新闻视频检索技术研究[J]. 计算机与数字工程，34(10)：83-86

[3] 董献洲，胡晓峰，吴琳，等. 虚拟新闻的表达与生成及其系统设计与实现[J]. 系统仿真学报，2006，18(12)：3634-3636

[4] 陈芳莉，胡晓峰，吴琳，等. 虚拟新闻模拟系统的研究与设计[J]. 计算机仿真，2007，24(8)：5-7

[5] 罗卫光. 电视深度报道的叙事学研究[D]. 广州：暨南大学，2004

[6] 张军华，王晓勇. 电视新闻叙事的视角转换与主题建构[J]. 广西师范大学学报：哲学社会科学版，2005，41(3)：59-61

[7] 司光亚，胡晓峰，吴琳. “沉浸式”战略决策训练模拟系统研究与实现[J]. 系统仿真学报，2006，18(12)：3581-3583

[8] 卢亮，张博文. 搜索引擎原理、实践和应用[M]. 北京：电子工业出版社，2005

[9] 孙宾. 现代汉语文本的词语切分技术[R]. 北京：北京大学计算语言学研究所，2003

[10] 苗夺谦，卫志华. 中文文本信息处理的原理与应用[M]. 北京：清华大学出版社，2007