文章编号:1001-2486(2006)06-0114-06

多自治空间数据源的 k 邻近查询处理*

唐桂芬,刘书雷,廖 巍,景 宁,陈 荦
(国防科技大学 电子科学与工程学院,湖南 长沙 410073)

摘 要:多自治空间数据源的 k 邻近查询处理在空间信息领域具有广泛的应用。综合分析比较现有的查询处理算法,研究并提出了多自治空间数据源环境下 k 邻近查询处理框架及其实现算法。实验结果表明,所提出的算法能有效地减少 k 邻近查询处理系统的数据传输量,减少了系统响应时间。

关键词:自治空间数据源;k 邻近查询;数据源 R 树索引
中图分类号:TP392 **文献标识码**:A

k-NN Query Processing for Multi-autonomous Spatial Data Sources

TANG Gui-fen, LIU Shu-lei, LIAO Wei, JING Ning, CHEN Luo
(College of Electronic Science and Engineering, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: k-NN query based on multi-autonomous spatial data source is widely used in spatial information realm. In this paper, incorporating existing k-NN query algorithm, a new k-NN query framework for multi-autonomous spatial data sources and its implement algorithm, is proposed. The experiment results show that the method performs well both in data transmission volume and corresponding time of the k-NN query system.

Key words: autonomous spatial data source; k-NN query; data source R-tree index

自治空间数据源(ASDS, Autonomous Spatial Data Source)是通过约定接口远程访问的空间数据库或空间数据文件,数据源内部数据的组织、管理以及接口实现均对用户透明。k 邻近[1](k-NN, k Nearest Neighbor)查询处理是指从数据源中检索出距查询点最近的 k 个对象的查询处理过程。面向多自治空间数据源的 k-NN 查询在空间信息领域具有广泛的应用[2]。

当前针对自治空间数据源的 k-NN 查询的研究还不多。Schwarz 等人[2]首先研究了多自治空间数据源 k-NN 查询处理问题,提出了迭代放大、缩小查询范围动态确定候选数据源的方法。Schwarz 没有采用任何索引结构,每个迭代过程都要遍历所有空间数据源,这种处理方法对数据源数目很大的情况是非常低效的。Liu 等人[3]提出了利用迭代窗口查询实现单个远程空间数据库的 k-NN 查询处理算法。在多自治空间数据源环境下,查询窗口与空间数据源位置关系非常复杂,Liu 等人所提出的算法不能适应这种复杂性。

本文研究并提出了一种高效的多自治空间数据源的 k-NN 查询处理方法。提出了过滤—查询—归并查询框架。在过滤阶段,提出了基于 R 树的空间数据源过滤算法快速过滤与查询结果无关的空间数据源;查询阶段,改进了 Liu 等人提出的 k-NN 查询处理算法,使其适应查询窗口与空间数据源位置的不确定性变化。实验表明,本文提出的算法能够有效减少数据传输量和缩短系统响应时间。

1 多自治空间数据源 k 邻近查询处理

首先给出多自治空间数据源的 k 邻近查询处理的定义。

定义 1 多自治空间数据源的 k 邻近查询处理,P 为给定查询点,k 为邻近点的个数,自治空间数据源集合 $S=\{ASDS_1, ASDS_2, \cdots, ASDS_n\}$,$\|(P,o)\|$ 表示查询点 P 到空间对象 o 的距离,则面向 S 的 k-NN 查询处理过程就是从 S 中查找出 k 个具有最近距离 $\|(P,o)\|$ 的对象,其中 $o\in\bigcup_{i=1}^{n} ASDS_i$,记

* 收稿日期:2006-05-14
基金项目:国家 863 高技术资助项目(2002AA134010,2002AA104220)
作者简介:唐桂芬(1977—),女,博士生。

为 $KNN4MS(P,k,S)$,则有:

$$KNN4MS(P,k,S)=\{(o_1,o_2,\cdots,o_k)\mid \|(P,o)\|\leqslant\|(P,o')\|,$$

$$where\ 1\leqslant i\leqslant k, o_i\in\bigcup_{i=1}^{n}ASDS_i,\ \forall o'\in S-\{o_1,o_2,\cdots,o_k\}\}$$

实现 $KNN4MS$ 处理需要解决以下问题:(a) 空间数据源的有效选取,可用的数据源大部分与查询结果无关,依据数据服务的服务能力快速过滤无关数据源,从而避免对无关数据源的查询处理;(b) 候选数据源的高效 k-NN 查询处理;(c) 将查询阶段所产生的候选中间数据对象归并为满足用户查询的结果。据此,我们将多自治空间数据源 k 邻近查询处理过程分解为三个步骤:即过滤、查询与归并三个阶段。

候选数据对象的归并本质上是一个本地 k-NN 查询处理,可利用现有的技术方法[2]。本文聚焦有效的空间数据源过滤方法和候选数据源 k-NN 查询处理方法。

2 空间数据源过滤算法(FilterSDS)

首先将每个空间数据源作为一个独立的对象存储在 R 树的叶节点中,为空间数据源构建 R 树索引结构。该索引结构包含两类节点记录:非叶节点,其数据结构为 $<MBR, Child\text{-}Pointer>$,其中 MBR 是以该记录为根的子树的最小外包围矩形,$Child\text{-}Pointer$ 是指向其子树的指针;另一种节点类型为叶节点,其数据结构为 $<MBR, NUM, Metadata\text{-}Pointer>$,其中,$MBR$ 是包含该空间数据源的最小外包围矩形,NUM 是数据源中包含的空间数据对象总数,$Metadata\text{-}Pointer$ 是该空间数据源在注册中心中的唯一标识号,通过该标识号可在注册中心快速定位空间数据源的元数据。

为了快速过滤与查询结果无关的数据源,引入两种度量准则作为剪枝的依据,一种度量准则为查询点到空间数据源的最近距离($MinDistance$)和最远距离($MaxDistance$),$MinDistance$ 和 $MaxDistance$ 分别为查询点到空间数据源 MBR 的最小和最大欧式距离,其具体计算方法参见文献[1];另一种度量准则是查询点到候选空间数据源集合的最大距离,记为 $MaxD$,$MaxD$ 为候选数据源集合中各数据源 $MaxDistance$ 的最大值。据此我们有如下度量关系:

引理 1 对于给定查询点 P 和空间数据源集合 S,M 为 S 中的任一空间数据源,则有下述结论:

$$M.MinDistance\leqslant\|(P,o)\|\leqslant M.MaxDistance\leqslant MaxD,\quad where\ \forall o\in M,\forall M\in S$$

在空间数据源过滤阶段我们不能预测数据源内部数据对查询结果集的影响,剪枝过程中在考虑分枝上下界的同时还需要兼顾数据源中包含的对象数目。据此,数据源 R 树搜索剪枝策略可描述为:

(1) 若当前处理节点 N 有 $N.MinDistance$ 大于当前的 $MaxD$ 且候选数据源集合中包含空间对象数目大于等于 k,则裁减该节点。

(2) 当有新的空间数据源加入候选数据源集合时,考察候选数据源列表。M,N 分别为候选数据源列表中的两个不同的数据源,若 $N.MinDistance>M.MaxDistance$ 且数据源列表中 M 之前所有数据源包含的空间对象总数大于 k,则将 N 从候选数据源集合中删除。

图 1(a)为剪枝策略 1 示例,其中 k 等于 6。若当前候选空间数据源集合 S 为 $\{ASDS_3, ASDS_2\}$,则 $MaxD=ASDS_2.MaxDistance$。对于 $ASDS_4$ 和 $ASDS_1$,有 $MaxD<ASDS_4.MinDistance$,$MaxD<ASDS_1.MinDistance$,且 S 中包含的空间数据对象数目大于 6,则认为 $ASDS_4$ 和 $ASDS_1$ 与查询结果无关,无需进一步查询处理,裁减之。

图 1(b)为剪枝策略 2 示例。若当前候选数据源集合为 $\{ASDS_2, ASDS_4, ASDS_1\}$,此时 $MaxD=ASDS_1.MaxDistance$。考察数据源 3,有 $ASDS_3.MinDistance<MaxD$,则将 $ASDS_3$ 加入候选数据源列表,形成新的数据源集合 $\{ASDS_2, ASDS_4, ASDS_3, ASDS_1\}$。如图 1,考察候选数据源列表,发现 $ASDS_1.MinDistance<ASDS_4.MaxDistance$ 且 $ASDS_2$、$ASDS_3$ 和 $ASDS_4$ 包含空间对象数目大于 6,则可将数据源 $ASDS_1$ 从候选数据源列表中删除。剪枝策略 2 能防止因空间数据源 MBR 交叠导致候选数据源范围无限扩大。

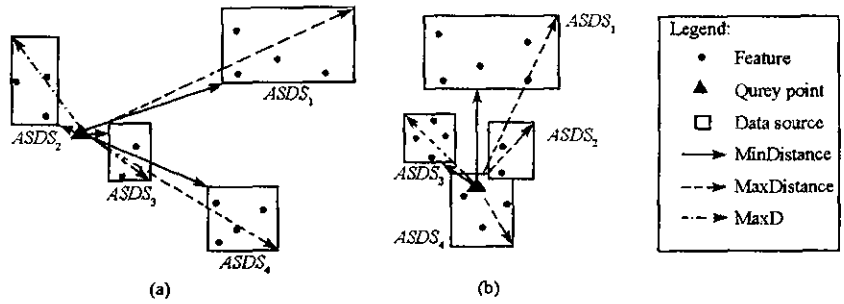

图1 2维空间剪枝策略示例

Fig. 1 Example for search pruning strategy in 2D

具体空间数据源过滤算法(FilterSDS)描述如下：

算法1 空间数据源过滤算法(FilterSDS)。

输入：查询点 P**，邻近点个数** k**，当前处理节点** $Node$**，当前候选数据源列表中数据对象总数** $TotalNum$**，查询点到候选数据源集合的最大距离** $MaxD$**。**

输出：候选数据源列表 ListDS[]**。**

1) $Node.MinDistance > MaxD$ **且** $TotalNum > k$**，则裁减该节点。**

2) **否则，**

2.1) **若** $Node$ **为叶节点，将该节点加入候选数据源集合；**

2.2.1) **对** $ListDS$[] **进行重新排序，记** $Node$ **在** $ListDS$[] **中位置为** t**，** $ListDS$[] **长度等于** l**；**

2.2.2) **用剪枝策略2调整** $ListDS$[]**；**

每个 $j \in [t,l]$**，相应的** $i \in (j,l]$**，若** $ListDS[i].MinDistance > ListDS[j].MaxDistance$**，且** $\sum_{m=1}^{j} ListDS[m].NumofObject > k$**，将** $ListDS[i]$ **从候选数据源集合中删除；**

2.2.3) **重新计算** $MaxD$ **和** $TotalNum$**。**

2.2) **若** $Node$ **为非叶节点，读取该节点的所有子节点，置入临时列表中。**

2.2.1) **计算各子节点** $MinDistance$ **和** $MaxDistance$**，并按** $MaxDistance$ **对临时列表排序；**

2.2.2) **递归调用空间数据源过滤算法处理临时列表中各节点，直至临时列表为空。**

3 候选空间数据源的 k 邻近查询处理算法(CSKNN)

本文所处理的空间数据源不直接提供邻近查询处理接口，不能直接向数据源提交 k **邻近查询请求。文献[2]研究了针对这类数据源的** k-NN **查询处理算法，其主要思想如图2所示，通过迭代估计使得最终查询窗口的** $range$ **值不小于第** k **个邻近点到查询点的距离，将** k-NN **查询转化为窗口查询。该算法的核心是** $range$ **的估计算法。文献[2]仅考虑查询点及查询窗口始终在数据源内部的情形。事实上还存在下面的情形，分别如图3(b)、3(c)所示：(1) 迭代过程中查询窗口与数据源** MBR **只有部分相交；(2) 查询点落在数据源外部。考虑到这些情形，我们引入实际查询窗口概念并对算法进行改进，实际查询窗口的具体定义如下：**

定义2 实际查询窗口(AQW**)，是查询窗口和空间数据源** MBR **相交的部分。如图3中，矩形Ⅰ为数据源** MBR**，矩形Ⅱ为查询窗口，而实际查询窗口为图中灰色部分。**

为了估计 $range$ **值，我们将实际查询窗口面积值域进行划分，构建查询窗口和数据源位置关系与实际查询窗口面积值域的映射。具体地说，以查询点到数据源** MBR **四个边垂直距离作为** $range$ **值，当查询窗口与数据源有交集时，计算相应的实际查询窗口面积值，从而实现对实际查询窗口值域的划分。例如，查询点与数据源位置关系如图3(a)所示，查询点到数据源各边的垂直距离依次为** a**，** b**，** c**，** d**，其中**

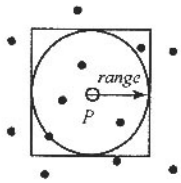

图 2 k-NN 查询示例
Fig. 2 Example for k-NN query

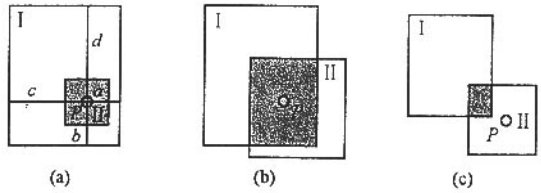

图 3 查询窗口与数据源 MBR 之间的位置关系
Fig. 3 Position relation of the query window and the MBR of data source

$a<b<c<d$。实际查询窗口面积值域划分为 $\{(0,a^2),(a^2,b^2),(b^2,c^2),(c^2,d^2),(d^2,\infty)\}$。若实际查询窗口面积 $s\in(0,a^2)$,则采用图 3(a)对应 $range$ 值求解方法,而 $s\in(a^2,b^2)$,则采用图 3(b)对应的 $range$ 值计算方法,依此类推。改进后的候选数据源 k-NN 查询算法描述如算法 2。

算法 2 改进的候选数据源的 k-NN 查询处理算法(CSKNN)。

输入:空间数据源中包含空间对象总数 $TotalNum$,查询点为 P,邻近点个数为 k。

输出:候选数据对象列表 $ObjectList[]$。

1) 若 $TotalNum>k$;

1.1) 计算查询点到数据源 MBR 各边的垂直距离,并计算相应的实际查询窗口面积值。据此对实际查询窗口面积值域进行划分;

1.2) 利用数据源的平均密度计算初始的实际查询窗口面积 $Area(AQW)_{first}=\dfrac{Area(MBR)}{TotalNum}\times k$;

1.3) 根据实际查询窗口面积确定查询窗口和数据源 MBR 的相交关系,依据相应的求解方法计算 $range$ 值,则 $QW\leftarrow\{(P.x-range,P.y-range),(P.x+range,P.y+range)\}$;

1.4)向数据源提交窗口查询操作,将查询结果放入 $ObjectList[]$。

1.4.1) 对任意的 $object\in ObjectList[]$ 若 $\|(P,object)\|>range$,则将该数据对象从候选数据对象列表中删除,候选数据对象数目 $Count$ 减 1;

1.4.2) 若 $Count<k$;

1.4.2.1) 若 $Count=0$,则直接将 $range$ 扩大一倍,确定新的查询窗口并跳转到 1.4);

1.4.2.2) 否则,利用 $Area(AQW)$ 和 $Count$ 计算当前实际查询窗口的平均密度,并据此估计新的实际查询窗口面积 $Area(AWQ)$,跳转到 1.3)。

1.4.3) 否则,保留 $ObjectList[]$ 作为候选数据对象,返回。

2) 否则,返回数据源中所有数据对象。

$CSKNN$ 算法中 1.4.1)保证候选数据对象都在查询点为中心,$range$ 为半径的圆域内;1.4.2)保证了 $TotalNum>k$ 时返回至少且接近 k 个候选数据对象。算法 2 能保证 k-NN 查询结果的完备性。

4 实验与讨论

我们在局域网内构造实验环境验证所提出的算法的性能。测试数据包括真实数据和合成数据。(1)合成空间数据源(SSDS, Synthetic Spatial Data Source) 均为点对象,数据对象总共 1 000 000 个,形成10 000个空间数据源;(2) 真实的空间数据源(RSDS, Real Spatial Data Source):从上海数字城市空间数据库中提取空间数据集,包含 185 817 个面对象,按行政区划划分以及要素类型划分到 200 个空间数据源中;测试数据分散在局域网内的 9 台服务器上。

实验中,数据源包装成标准的 WFS[4],使用 Oracle 数据库管理 WFS 的元数据,模拟注册中心。使用 SOAP 协议与 WFS 通信,中间结果集采用 GML[5] 文档形式存储和传输。每组实验均在数据空间内随机生成 100 个查询点,为每个查询点进行不同 k 值邻近查询处理,其中 $k\in\{1,5,10,15,20,25,50,100,500,1000\}$。

实验1:使用SSDS实验,评价FilterSDS算法和CSKNN算法的性能。

为了评价FilterSDS算法性能,考察不同 k 值FilterSDS算法平均调用数据源的个数,实验结果如图4,表明FilterSDS算法平均调用数据源个数随 k 值增加而增加。当 k 值较小时,FilterSDS算法过滤掉了绝大多数空间数据源,过滤效率非常高,而 k 值较大时数据源过滤效率相对较低。

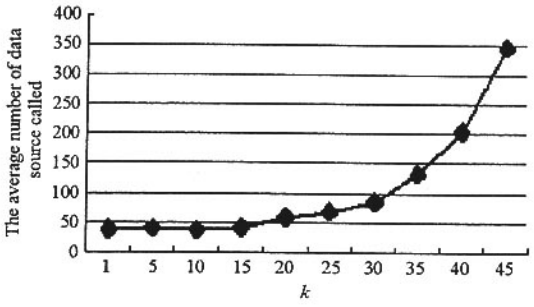

图4 平均调用的数据源个数

Fig. 4 Average data sources called in SSDS

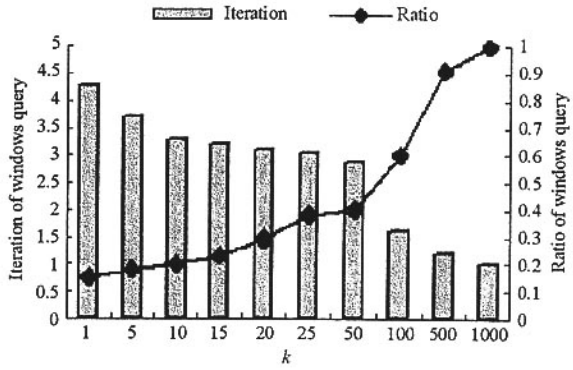

图5 窗口查询迭代次数和查询窗口比率

Fig. 5 Iterations and ratio of windows query in SSDS

引入两个指标评价CSKNN算法的性能,即窗口查询迭代次数和查询窗口比率。窗口查询迭代次数反映了算法的收敛效果。查询窗口比率是指获得最终查询结果时实际查询窗口的面积与数据源MBR面积的比值;查询窗口比率能很好地体现窗口查询从数据源中提取数据对象的效率。实验结果如图5所示。图5表明迭代次数随 k 值增加而减少,所以 k 值越大收敛效果越好;而查询窗口比率则随 k 值增大而增大,故 k 值越小提取过滤的效率越高。

实验2:为了评价FilterSDS算法和CSKNN算法对查询效果的影响,使用SSDS和RSDS分别实验,考察实验环境中下面三种 k-NN处理方法的整体性能:(1) FilterSDS算法和CSKNN算法都不使用,本地实例化所有数据,记为 $A1$;(2) 使用FilterSDS算法而不使用CSKNN算法,实例化候选数据源的全部数据,记为 $A2$;(3)同时使用FilterSDS算法和CSKNN算法,选择性实例化,记为 $A3$。

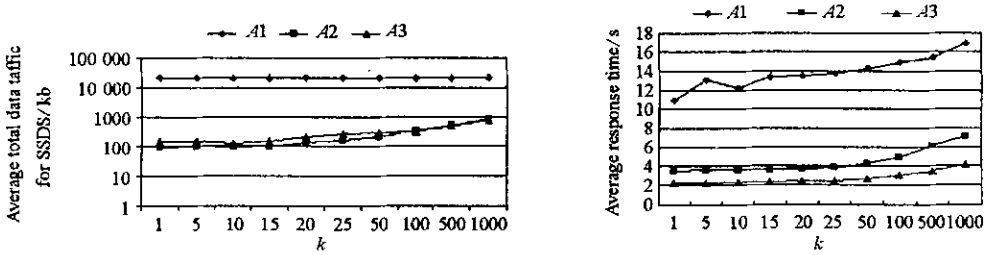

图6 (a) SSDS平均总的数据传输量;(b) SSDS的平均相应时间

Fig. 6 (a) Average total data traffic for SSDS;(b) Average response time for SSDS

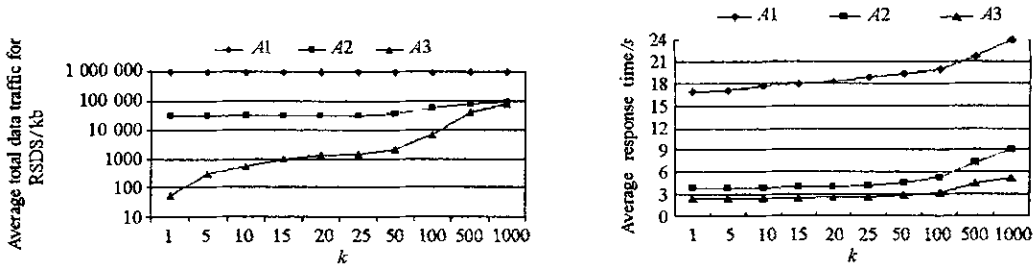

图7 (a) RSDS平均总的数据传输量;(b) RSDS的平均相应时间

Fig. 7 (a) Average total data traffic for RSDS;(b) Average response time for RSDS

选择平均总的数据传输量以及平均查询请求平均响应时间作为评价指标。比较 $A1$、$A2$ 和 $A3$ 的查询性能。实验结果表明FilterSDS算法和CSKNN算法减少查询系统数据传输量并减少了系统查询

时间。图 6 中,当 k 值较小时,$A3$ 和 $A2$ 比较并无优势,而图 7 中则表明 $A3$ 较 $A2$ 有较大的性能提高。k 值较小时,候选数据对象数目较少,同时合成数据均为点对象,查询结果占整个响应消息的比重较小。而在真实数据的数据类型均为复杂的面对象,每个对象的数据量都很大,实际的数据量相对整个响应消息中比重较大,所以有图 7 中的大幅性能提高。而 k 值较大时,候选数据源数目较大,查询窗口比率较大,$A3$ 较 $A2$ 和 $A1$ 尤其是 $A2$ 的性能优势不大。所以 FilterSDS 算法对 k 值较小时能更好地减少数据传输量。

图 6(b)和图 7(b)的实验表明,无论是真实数据还是合成数据,FilterSDS 算法过滤了大量与查询结果无关的空间数据源,同时 CSKNN 算法进一步减少了候选数据对象的数量,减少数据传输的时间开销;中间服务器上候选邻近对象数目的减少也缩短了中间服务器上归并处理的时间。

参 考 文 献:

[1] Roussopoulos N, Kelley S, Vincent F. Nearest Neighbor Queries[A]. In Proc. of SIGMOD[C], 1995.

[2] Schwarz T, Iofcea M, Grossmann M, et al. On Efficiently Processing Nearest Neighbor Queries in a Loosely Coupled Set of Data Sources [A]. In the Proc. of ACMGIS[C], 2004.

[3] Liu D, Lim E, Ng W. Efficient k Nearest Neighbor Queries on Remote Spatial Databases Using Range Estimation[A]. In Proc. of SSDBM[C], 2002.

[4] Open GIS Consortium. Web Feature Service Implementation Specification, Version :1.1.0. OpenGIS Implementation Specification[S]. http://www.opengis.org/docs/04-094.pdf, 2005-05-03.

[5] Open GIS Consortium. Geography Markup Language (GML) 2.0, OGC Recommendation Paper[S]. http://www.opengis.org/docs/01-029.pdf, 2001-02-20.

(上接第 75 页)

选择模平方后的积分宽度(W)也是该算法应慎重考虑的问题。本文以保证直扩信号在频率搜索区间内的检测性能为依据来选择该参数。若选定的 W 满足 PM 支路的检测性能要求,则 PM 支路仅以当前积分数据作单次判决;否则可利用该支路之前获得的数据进行序贯检测来提高其检测性能。

3 结束语

本文提出的处理结构已经在 Xilinx 公司器件 Virtex-II 3000 上实现。如果仅从伪码捕获的角度考虑,该算法相对滑动相关法等捕获方案在缩短捕获时间的同时增加了系统的硬件规模,但从总体上说硬件规模反而减小了,这是因为:(1)FFT/IFFT 可以复用处理;(2)PM 模式识别与捕获可复用 FFT 单元,只需增加少量硬件资源就可实现 PM 模式的宽带快捕;(3)抗干扰处理可复用 FFT/IFFT 单元,也只需增加少量硬件资源就可以实现直扩模式下的窄带干扰抑制处理。

参 考 文 献:

[1] Kwon H M. Third-generation TDRSS-compatible Direct-sequence Spread-spectrum Digital Receiver[J]. IEEE Transactions on Vehicular Technology, 1997,46(4).

[2] 郭贵堂,等. TDRSS 系统 S 波段双模式用户应答机[J]. 飞行器测控学报, 2002,21(4).

[3] 李秉尚. TDRSS、USB 系统兼容星载应答机的一种实现方案[J]. 飞行器测控学报, 2001,20(3).

[4] 王诺,等. TDRSS 中频信号捕获与跟踪的数字化实现[J]. 无线通信技术,2002(4).

[5] 王诺,等. 改进的数字化 TDRSS 中频信号捕获跟踪系统[J]. 通信学报, 2003,24(6).

[6] 郑林华,等. 星用 DS/PM 双模式应答机的设计与实现[J]. 国防科技大学学报, 2004,26(4).

[7] Anonymity. Performance and Design Requirements and Specification for the Fourth Generation TDRSS User Transponder[R]. National Aeronautics and Space Administration,1996.

[8] Liu Y F, Chen Z J. Implement and Performance Analysis of PN Codeacquisition Based on FFT[Z]. Fifth World Congress on Intelligent Control and Automation, Hangzhou, 2004.