Improving Spatial neighbor Index Performance Based on Space-Filling Curves

Nzar Abdulqader Ali *Payman Othman Rahem*
College of Administration - University Of Sulaimani
nzar@mail.com
Received date: 2/10/2013 Accepted date: 27/2/2014

Abstract

Spatial data consists of objects in space made up of points, lines, regions and data of higher dimensions. Access method is required to support efficient manipulation of the multi-dimensional spatial objects in the secondary storage. The goal of the Space-Filling Curve (SFC) is to preserve spatial proximity; they can handle Nearest Neighbor Queries (NNQ) which involves determining the point in a dataset that is nearest to a given point. In this paper a new algorithm for finding the horizontal and vertical neighbor for RBG curve is proposed. The four direction neighbors are directly founded from the query block without depending on transformation method between Piano and RBG index. The result shows that the new algorithm has better performance than the traditional RBG neighbor index finding by reducing the time needed for transformation between RBG and Piano index.

Keywords: **Spatial Database, Spatial Access Methods, Space Filling Curves, Nearest Neighbor Queries.**

تحسين فهرسة الأداء في قواعد البيانات المكانية بأستخدام منحنى ملئ الفضاء

نزار عبدالقادر علي بيمان عثمان رحيم
كلية الادارة والاقتصاد . جامعة السليمانية
تاريخ استلام البحث:2012/4/2 تاريخ قبول البحث:2014/2/27

الخلاصة

البيانات المكانية تحتوي على عدد كبير من المكونات المكانية مثل النقاط، الخطوط و متعدد الأضلاع (المناطق). ان طريقة الوصول الى بيانات متعددة الأبعاد تعتبر اكثر تعقيدا مقارنة ببيانات تتكون من بعد واحد وذلك بسبب عدم امكانية ايجاد الموقع المكاني المحفوظ. لذا تحتاج لعملية تحويل من فضاء متعدد الابعاد الى فضاء ذو بعد واحد، بحيث تكون المكونات القريبة في الفضاء تحول الى نقاط متقاربة على الخط لحفظ الموقع المكاني. ان منحنى ملىء الفضاء (Space–Filling Curve) يمر خلال كل نقطة في الفضاء مرة واحدة. وذلك لضمان الهدف الا وهي الحفاظ على المكان التقريبي، لذا من الممكن معالجة استعلامات المجاور الاقرب (Nearest Neighbor Queries) والتي تتضمن ايجاد النقطة الاقرب الى نقطة الاستعلام (Query Point). بعض الامثلة على منحنيات ملىء الفضاء هي منحنى (Peano) و منحنى (RBG). عند ترتيب البيانات خطيا باستخدام منحنى (Peano) يمكن مباشرة ايجاد الاتجاهات الأربعة للاستعلام اعتمادا على خاصية الوحدة الثنائية. لكن عند ترتيب البيانات باستخدام منحنى (RBG) ايجاد المجاور يكون اكثر تعقيدا. في هذا البحث، تم اشتقاق خوارزمية جديدة لأيجاد الأتجاهات الأربعة لأستعلام منحنى (RBG) مباشرة دون الحاجة الى التحويل بين المنحنيين.
ان الاستنتاجات التي تم التوصل اليه تبين بان الخوارزمية المقترحة تؤدي الى تقليل زمن تنفيذ الاستعلام وذلك بعد التخلص من كلفة زمن التحويل بين منحنى (RBG) ومنحنى (Peano). وختاما تم مقارنة الاداء الفهرسي (Index) لاحجام بيانات مختلفة ولاعماق (Depth) مختلفة

الكلمات الدالة : قواعد البيانات المكانية ، طريقة الوصول المكاني ، منحنى ملىء الفضاء، استعلامات المجاور الاقرب .

Introduction

The role of spatial database is continuously in incensement in many modern applications during the last few years. Mapping, urban planning, transportation planning, resources management, Geographic Information System (GIS), Computer Aided Design (CAD) are some of these applications.

The key characteristic that makes a spatial database (SDB) a powerful tool is its ability to manipulate spatial data, rather than simply to store and represent them. The most basic form of such a manipulation is answering queries related to the spatial properties of data. Some typical spatial queries are the following:

- "Point Query" seeks for the spatial objects that fall on a given point.
- "Range Query" seeks for the spatial objects that are contained within a given region.
- "Join Query" may take many forms. It involves two or more spatial data sets and discovers pairs of spatial objects that satisfy a given predicate.

Finally, a very common spatial query is the "Nearest Neighbor Query" that seeks for the spatial objects residing more closely to a given object. In its simplest form, it discovers K Nearest Neighbors (K-NN) for a given object [1].

Database systems that support spatial data use spatial indexes, also known as Spatial Access Methods (SAM), to speed up the response of the query time. Because of the large volume of spatial objects, spatial access methods are usually used to organize and speed up the retrieval of spatial objects. Several spatial as the access methods make use of a regular decomposition of space in order to organize and store spatial data. A spatial indexing method organizes space and the objects in it in some way so that the only parts of the space and a subset of the objects need to be considered to answer a query. In order to find spatial objects efficiently based on proximity, it is essential to have an index over spatial locations. The underlying data structure must support efficient spatial operations, such as locating the neighbors of an object and identifying objects in a defined query region [2].

Most indexes are based on the principle of divide and conquer. Indexing structures following this approach are typically hierarchical. The approach is naturally suitable for a database system where the memory space is limited, and hence the pruning of a search must be performed such that the more detail to be examined, the smaller numbers of objects are being examined. An advantage of hierarchical structures is that they are efficient in range searching.

Indexing in a Spatial Database is different from indexing in a conventional database in that data in an SDB are multi-dimensional objects and are associated with spatial coordinates. The search is based not on the attribute values but on the spatial properties of objects [3].

Space Filling Curves (SFCs) provides a natural mapping from a high-dimensional space to a one-dimensional curve and the ordering of points on SFC has been used extensively as a meaningful order of points. In SFC the main index structures is Peano index and the other RBG and Hilbert index are constructed from the Peano index by transformation method proposed by Chen and Chang [4]. The nearest neighbor queries on the other hand used to find the nearest block to the query block depending on the same transformation method. In this paper new neighbor index structure proposed for RBG curve to find the nearest neighbor index (the four near block) from the query index block directly without depending on transformation method and then improve the performance of the query by reducing the transformation time. A program have been designed and developed to evaluate the performance improvement due to using the proposed algorithm.

SPACE-FILLING CURVES

A space-filling curve is a continuous path which passes through every point in a space once to form a one-one correspondence between the coordinates of the points and the one-dimensional sequence numbers of the points on the curve. The space-filling curve provides a way to order linearly the points of a grid. The goal is to preserve the distance, i.e., points which are close in space and represent similar data should be stored close together in the linear order. Some example of space-filling curves is the Peano curve, the RBG curve and the Hilbert curve.

In general, space-filling curves start with a basic path on a k-dimensional square grid of side 2. The path visits every point in the grid exactly once without crossing itself. It has two free ends which may be joined with other paths. The basic curve is said to be of order 1. To drive a curve of order *i*, each vertex of the basic curve is replaced by the curve of order *i*-1, which may be appropriately rotated and/or reflected to fit the new curve.

A. Peano Curve

The Peano curve is one-dimensional sequence number (1D-number) of points obtained by simply interleaving the bits of a binary representation of the X and Y coordinates of the point in the two-dimensional space (2D-space). The basic Peano curve for a 2x2 grid, denoted as P_1, is shown in Figure 1-(a). To drive higher orders of the Peano curve, we replace each vertex of the basic curve with the previous order curve. Figures 1- (b) and (c) show the Peano curves of

order 2 and 3, respectively. We can think of dividing the given region into quadrants and drawing a curve such as Figure 1-(a). Then each quadrant is divided in turn into 4 sub-quadrants, and the same basic curve repeated in each, in place of each node in the previous step. One more recursive step, again dividing each node into 4 sub-quadrants joined by the basic curve, gives rise to Figure 1-(c).

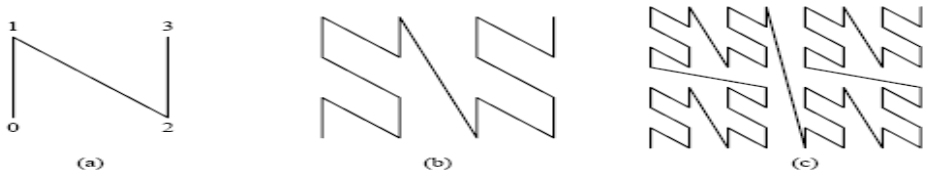


Figure 1: Peano curves of order: (a) 1; (b) 2; (c) 3

B. RBG Curve

In RBG, numbers are coded into binary representations such that successive numbers differ in exactly one bit position. Faloutsos [5] observed that difference in only one bit position had a relationship with locality. He proposed that numbers produces by interleaving the coordinates of points in 2D-space as in the peano curve technique to obtain the 1D-number. The basic reflected RBG curve of a 2x2 grid, denoted as R_1 is shown in Figure 2(a). The procedure to drive higher orders of this curve is to reflect the previous order curve over *x*-axis and then over the *y*-axis. Figure 2-(b) and (c) show the reflected binary curve of order 2 and 3, respectively. As in the case of Peano curve, the RBG curve begins with the curve of figure 2-(a) it divides each quadrant into 4 sub-quadrants and replicates. While replicating, it rotates the two upper quadrants through 180^o as shown in Figure 2-(b). It divides into 4 sub-quadrants once again, with replication and upper quadrant rotation to get Figure 2-(c).

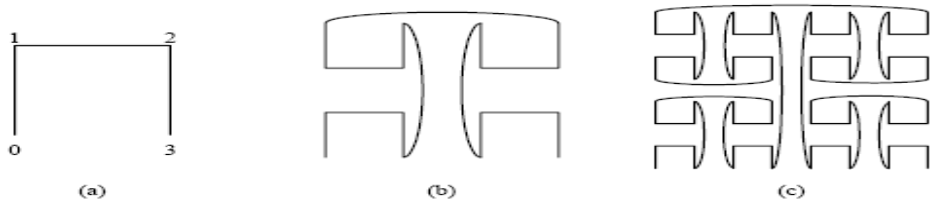


Figure2: RBG curves of order: (a) 1; (b) 2; (c) 3

C. Hilbert Curve

The Hilbert curve is mapping in which the four nearest neighbors in 2D-space are usually mapped to points not too far away in the linear traversal. It begins with Figure 3-(a). As in the case of the previous curves, it replicates in four quadrants. When replicating, the lower left quadrant is rotated clockwise 90^o, and the sense (or direction of traversal) of both lower

quadrants is reversed. The two upper quadrants have no rotation and no change of sense. Thus we obtain Figure 3-(b). Remembering that all rotation and sense computations are relative to previously obtained rotation and sense in a particular quadrant, a repetition of this step gives rise to Figure 3-(c). So the basic Hilbert curve of a 2x2 grid, denoted as H_1 is shown in Figure 3-(a). The procedure to drive higher orders of this curve is to rotate and reflect the curve at vertex 0 and at vertex 3. The curve can keep growing recursively by following the same rotation and reflection pattern at each vertex of the basic curve. Figure 3-(b) and (c) show the Hilbert curves of order 2 and 3, respectively.

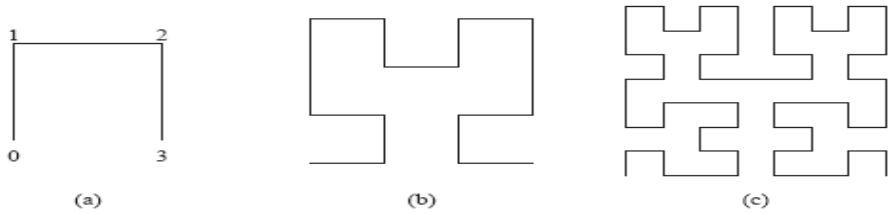
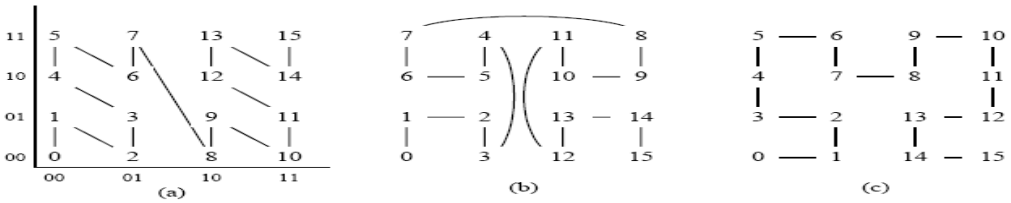


Figure 3: Hilbert curves of order: (a) 1; (b) 2; (c) 3

The path of space-filling curve imposes a linear ordering, which may be calculated at one end of the curve and following the path to the other end. Orenstein [6] used the term *z*-ordering to refer to the ordering of the Peano curve (Figure 3-(a)) he also used the term *z*-value to refer to the order of the point in the *z*-ordering. Similarly, in Figure 4-(b), *r*-ordering and *r*-value are used for the RBG curve, and in Figure 4-(c), *h*-ordering and *h*-value are used for the Hilbert curve.



Figher 4: Space Filling Curves of order 2: (a) Peano; (b) RBG, (c) Hilbert

PROPOSED ALGORITHM

Nearest neighbor-finding is one of the most important spatial queries in the field of spatial data structures which are concerned with proximity. Because the goal of the space-filling curves is to preserve spatial proximity, nearest neighbor queries can be handled by the space-filling curves.

Chen and Chang [CC05], finds the strategy based on the Peano curve for the nearest neighbor query. Next, they presented the rules for transformation between the Peano curve and

the other two curves, including the RBG curve and the Hilbert curve; also efficiently they fined the nearest neighbor by the strategies based on these two curves. Finally, they have compared the CPU_time and the I/O_time among different nearest neighbor strategies.

The proposed algorithm in this paper helps looking for the neighbor in horizontal and vertical direction for RBG curve directly without depending on transformation method from Peano curve and then decrease the execution time needed for transformation algorithm and as a result improve the performance of finding nearest neighbor of RBG curve.

Given a query point with its located block at the nth level (d_i) and the binary location code denoted as $\{x_i y_i\}$, $i = 0, 1, \ldots\ldots, n$. Let $k > 0$ be a positive integer and c be the numbers of ones in binary value $(R)_2$ Figure 5 shows the block diagram of the proposed algorithm .

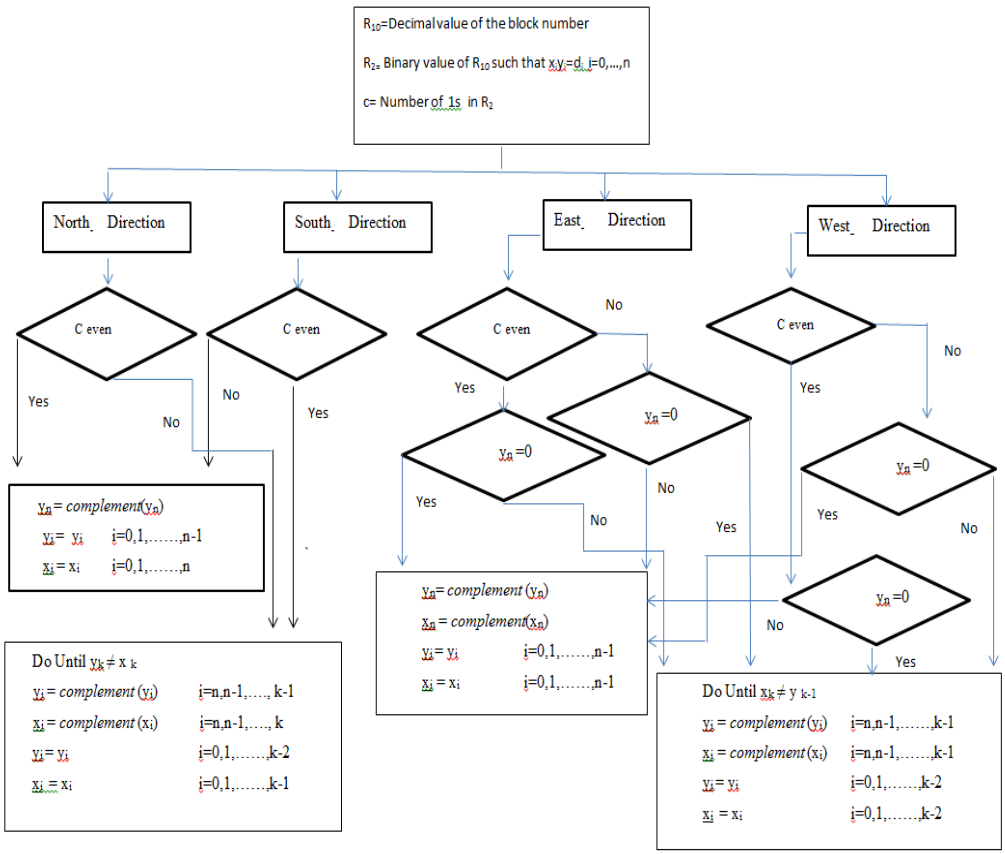


Figure 5: Block diagram for finding horizontal and vertical direction for RBG curve

Example:

Figure 6 represents RBG curve of order 3 as decimal values. We implement our algorithm depending on the equations represented in the block diagram to find the horizontal and vertical neighbor of block 323.

133	130	103	100	233	230	203	200
132	131	102	101	232	231	202	201
121	122	111	112	221	222	211	212
120	123	110	113	220	223	210	213
013	010	023	020	313	310	**323**	320
012	011	022	021	312	311	322	321
001	002	031	032	301	302	331	332
000	003	030	033	300	303	330	333

Figure 6: The sequence numbers linearly orderd by the RBG curves

1) **North Neighbor.**

To find the north neighbor in vertical direction of block $(323)_{10}$, its binary form is $(x_0y_0, x_1y_1, x_2y_2)= (11,10,11)_2$, where c (number of 1s) =5 is odd and k=1 ($y_1 \neq x_1$), by changing the digits (y_0, x_1, y_1, x_2, y_2) according to equations represented in the block diagram (Figure 5), the north neighbor $10^*,0^*1^*,0^*0^*)_2 = (210)_{10}$ is obtained, the symbol * means the complement (change) bit.

2) **South Neighbor.**

To find the south neighbor in vertical direction of block $(323)_{10}$, its binary form is $(x_0y_0, x_1y_1, x_2y_2)= (11,10,11)_2$, where c (number of 1s) =5 is odd by changing the digit $y_{n=} y_2$ according to equations represented in the block diagram (Figure 5), the south neighbor $(11,10,10^*)_2 = (322)_{10}$ is obtained, the symbol * means the complement (change) bit.

3) **East Neighbor.**

To find the east neighbor in horizontal direction of block $(323)_{10}$, its binary form is $(x_0y_0, x_1y_1, x_2y_2)= (11,10,11)_2$, where c (number of 1s) =5 is odd and $y_n=y_2=1$, by changing the digits $x_{n=}x_2$ and $y_{n=}y_2$ according to equations represented in the block diagram (Figure 5), the east neighbor $(11,10,0^*0^*)_2 = (320)_{10}$ is obtained, the symbol * means the complement (change) bit.

4) **West Neighbor.**

To find the east neighbor in horizontal direction of block $(323)_{10}$, its binary form is $(x_0y_0, x_1y_1, x_2y_2)= (11,10,11)_2$, where c (number of 1s) =5 is odd, $y_n=y_2=1$ and k=2 ($y_1 \neq x_2$), by changing the digits (x_1, y_1, x_2, $y_{2)}$ according to equations represented in the block diagram

(Figure 5), the east neighbor $(11,0^*1^*,0^*0^*)_2 = (310)_{10}$ is obtained, the symbol * means the complement (change) bit.

PEFORMANCE MEASURMENT

The performance of the nearest neighbor-finding structure is compared based on three Space Filling Curves with new algorithm (NRBG) curve. The programs were created using visual basic language as the front end and SQL Server database as a backend database, the system depends on one of the important SQL function, which is CPU_BUSY time to calculate the system performance. The CPU_BUSY time returns the time that the CPU has spent working in milliseconds since Microsoft SQL Server was last started.

Figure 7 shows the grid decomposition produced by the program to find all spatial objects that are far away 50 km from the query point, where 50000 points of spatial objects are generated randomly and distributed over the sulaimani governorate map.

Figure 8 shows the cost of CPU_BUSY time for four distance length (25Km, 50Km, 75Km, and 100 Km) and for different depths to 50000 spatial objects. The results shows that the CPU time request for finding nearest neighbor depending on NRBG curve is less than RBG curve after finding the horizontal and vertical direction directly from query block without depending on transformation method and as a result the NRBG performance increased in comparison with RBG .

Figure 9 shows the cost of CPU_BUSY time for four distance length (25Km, 50Km, 75Km, and 100 Km) and for different depths to 150000 spatial objects. The results shows that the CPU time request for finding nearest neighbor depending on NRBG curve is still less than RBG which is means the algorithm give the same performance after increasing the size of spatial objects from 50000 to 150000.

From the Figure 10, by increasing the size of spatial objects to 250000, the NRBG curve have the same previous stability in comparison with RBG, and this represents the efficiency of the new algorithm for finding the neighbor points depending on the index of the neighbor node directly.

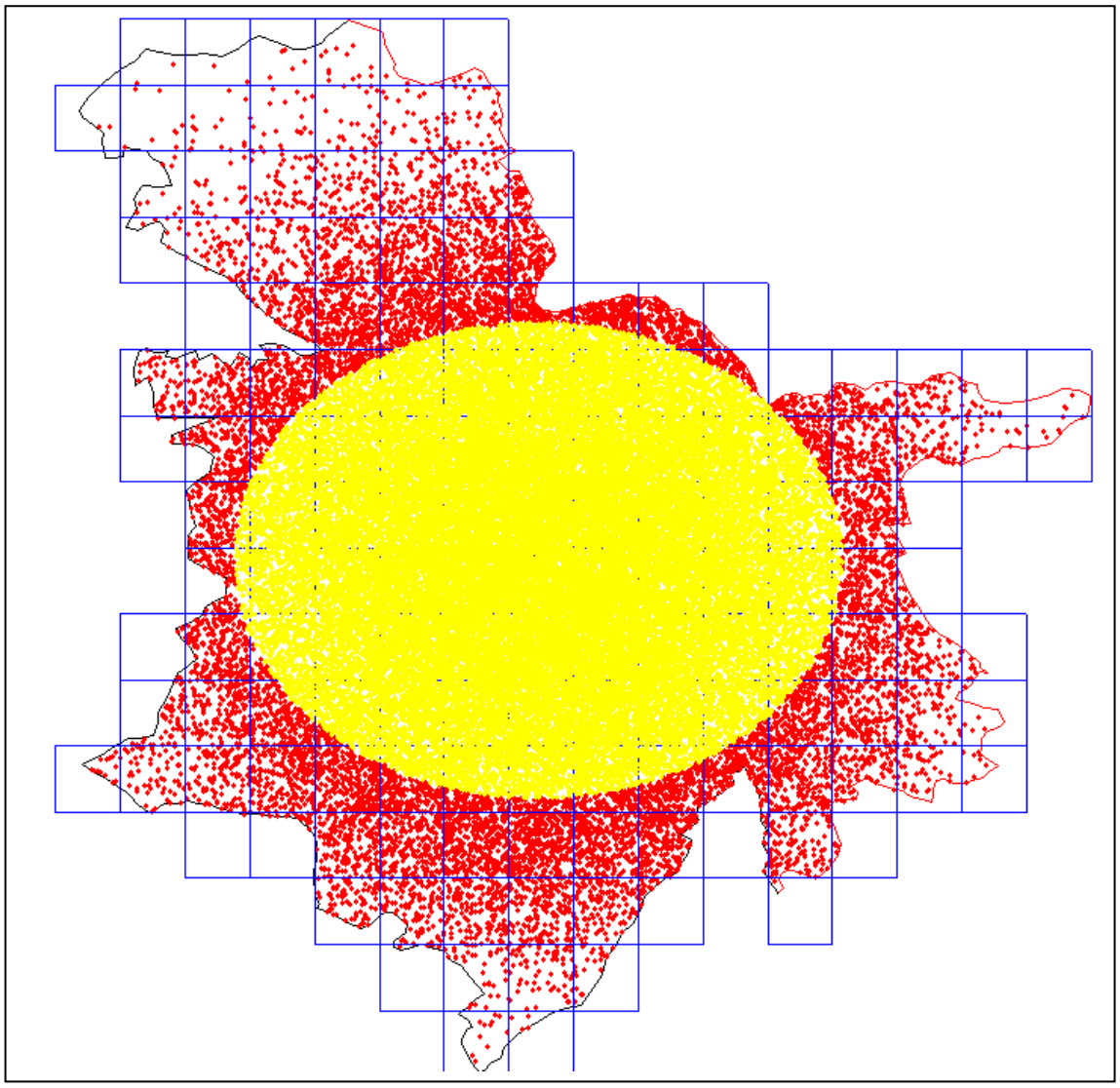
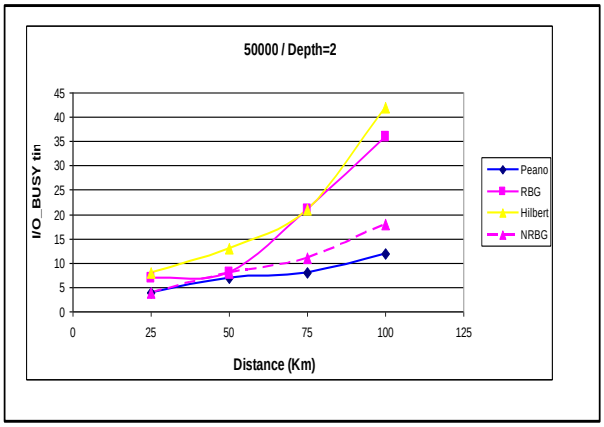
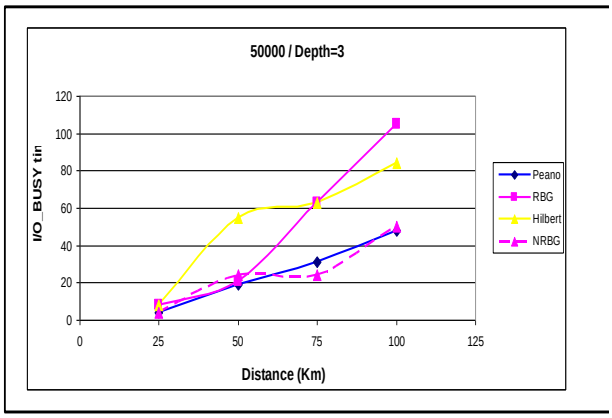


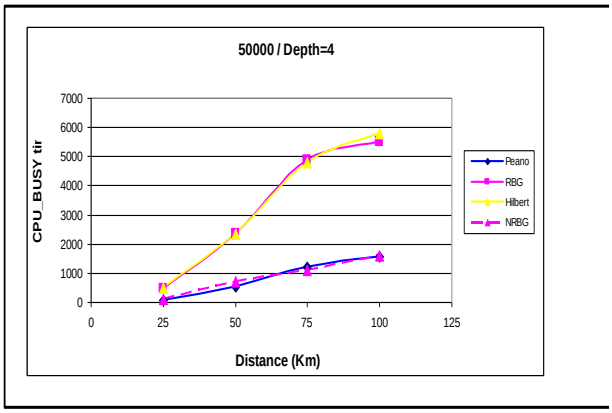
Figure 7: Query result for nearest neighbor for distance=50 km, depth=4



(a) Depth=2

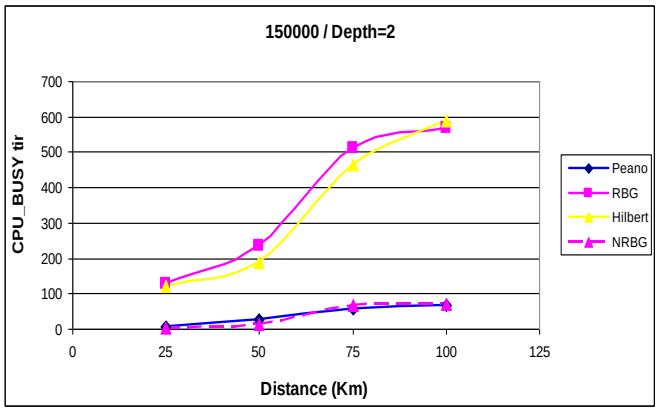


(b) Depth=3

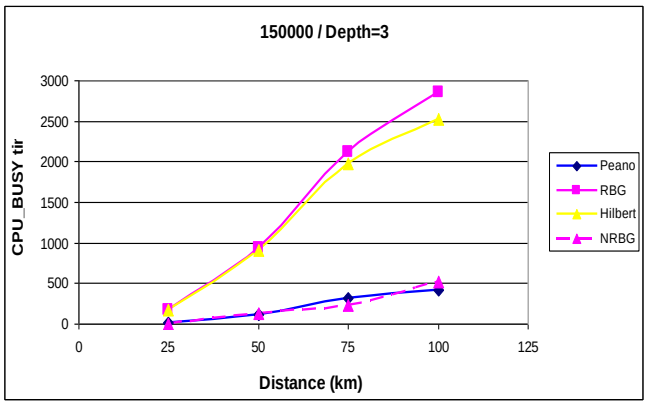


(c) Depth=4

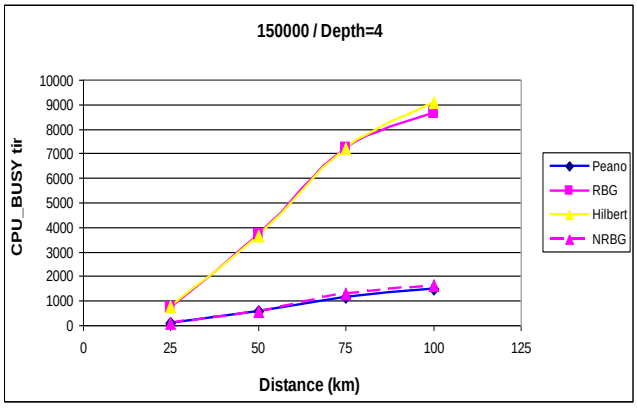
Figure 8: CPU_Busy Time for 50000 objects object for different depths



(a) Depth=2

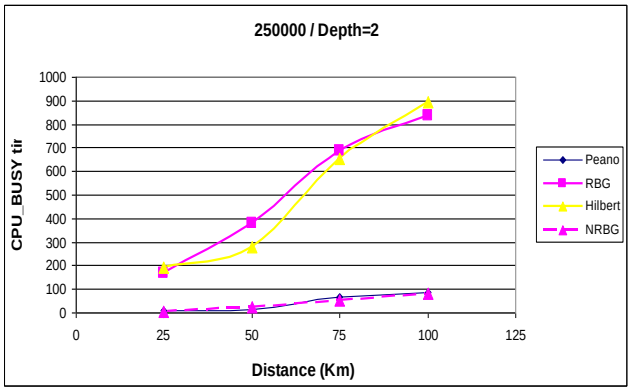


(b) Depth =3

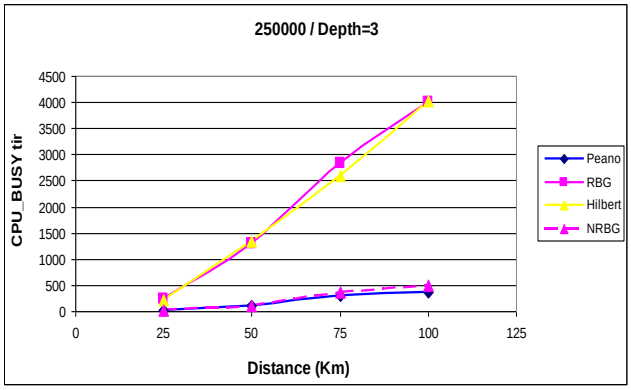


(c) Depth=4

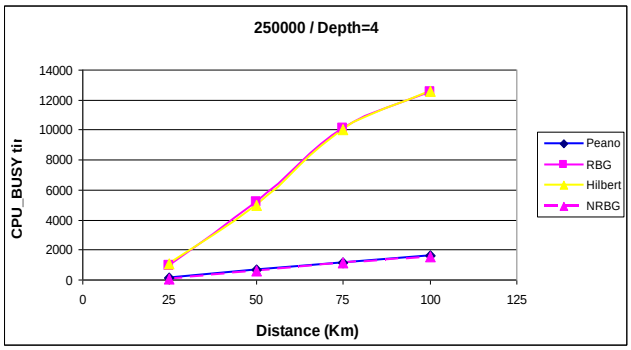
Figure 9: CPU_Busy Time for 150000 objects for different depths



(a) Depth =2



(b) Depth =3



(c) Depth =4

Figure 10: Comparison of CPU_Busy Time for 250000 objects for different depths

CONCLUSIONS

In this paper a comparison between three different space-filling curves is studied and a new algorithm (NRBG) for improving nearest neighbor query for RBG curve is proposed. The performance of the algorithm was calculated in terms of number of nodes visited to reach the nodes located within specified distance. It is observed that the proposed algorithm (NRBG) for finding NNQ gives better results than RBG and Hilbert curve and it is very close to Piano curve by excluding the transformation time. The results shows that the performance of NRBG curves remain stable after increasing the size of spatial objects from 50000 to 250000. The experimental results show that the system remains efficient for different depth size.

REFERENCES

[1] A. Corral, Y. Manolopoulos, Y. Theodoridis and M. Vassilakopoulos, "**Closest pair queries in spatial databases**" In Proceeding ACM SIGMOD Conference,2000,pp.189-200.

[2] W. G. Aref, H. Samet, "**A Window Retrieval Algorihm for Spatial database using Quadtree**", ACM-GIS, 1995, pp.69–76.

[3] B. Ooi, R.Davis and J. Han, "**Indexing in Spatial Databases**", 1993, Unpublished Manuscript available at: http://www.iscs.nus.edu.sg/˜ooibc.

[4] H. Chen, Y. Chang, " **Neighbor-Finding Based on Space-Filling Curves**", Information Systems, Vol 30, pp. 205-226, May 2005.

[5] C. Faloutsos, " **Multiattribute hashing using Gray Code**", In Proceeding ACM SIGMOD Conference,1986,pp.227-238.

[6] J.Orenstein, F. Manola, " **PROBE Spatial Data Modeling and Query Processing in an Image Database Application**", IEEE Trans. On Software Engineering ,pp.611-629,1988