# Efficient Range Query Using Multiple Hilbert Curves

**Chapter** · July 2011

DOI: 10.5772/22413 · Source: InTech



**1 author:**

Jing Dai
Virginia Polytechnic Institute and State University
**29** PUBLICATIONS **217** CITATIONS

# 19

# Efficient Range Query Using Multiple Hilbert Curves

Ying Jin[1], Jing Dai[2] and Chang-Tien Lu[3]
[1]*Cold Spring Harbor Lab*
[2]*IBM T. J. Watson Research Center*
[3]*Virginia Polytechnic Institute and State University*
*USA*

## 1. Introduction

Indoor location tracking based on RFID has been widely discussed and applied. RFID reading process is efficient and reliable, therefore it is suitable for discovering locations inside buildings where GPS signals are usually unreachable. In general, there are two approaches for location sensing using RFID. 1) Deploying RFID tags at fixed locations and RFID readers attached to moving objects (Willis, 2004). Each tag represents a reference point in the space, and a reader determines its location by the set of tags being detected. 2) Deploying RFID readers (and tags) at fixed locations and RDIF tags attached to moving objects (Hightower, 2001; Ni, 2004). The readers report to the system when a tag is detected, and the system identifies the location of this tag by the set of readers that have reported and their corresponding signal strength.

These location management systems require multi-dimensional access methods to allow efficient handling of spatial queries. Because there is no total ordering of locations that preserves the spatial locality between objects, it is difficult to design multi-dimensional access method in the way as traditional one-dimensional access methods. However, mapping multi-dimensional data into a single dimension makes it possible to utilize the extensively exploited B/B+-tree as the index and its associated concurrency control and recovery mechanisms.

Space-filling curves (SFCs) (Simmons, 1963) have been widely used to map the multi-dimensional data points into a linear order. It was first introduced by Peano (Peano, 1890) to map from a unit interval to a unit square. SFC can link all cells with passing through each of them only once, so it provides a way of generating a total linear ordering of all grids in a multi-dimensional space. Many SFCs have been proposed in the literature, such as Peano curve (or Z-order) (Orenstein & Merrett, 1984; Peano, 1890), Hilbert curve (Hilbert, 1891), Gray curve (Gray, 1953), Sweep, and Scan. The multi-dimensional data are transformed to a set of one-dimensional integer values using SFC mapping schemes. The transformed data can be stored in a traditional one-dimensional database based on the linear orders, and indexed by B-trees or B+-trees. Then the spatial queries, such as range query, kNN query, and spatial join, can be processed. Using SFCs to enable processing spatial queries based on traditional one-dimensional indices is proposed in (Faloutsos, 1988; Faloutsos & Rong, 1991; Faloutsos & Roseman, 1989; Jagadish, 1990).

Spatial range query identifies spatial objects located within a given area. For example, "find all hospitals in city A" is a common range query for a GIS application. In data mining applications, range queries are used to discover the characteristics of a specific region. For instance, a data set contains different environmental variables of the areas which are the habitat of some kinds of bird. A user may submit a range query like "find how many lakes or rivers within that area", in order to identify associations between water and bird. In addition, many spatial query operations, like k-nearest neighbor query, and spatial join query, rely on range queries. Developing an efficient range query processing scheme will contribute to improving overall spatial query operations.

Several works have been conducted on utilizing SFCs to solve range queries. Gray and Hilbert SFCs are used to handle range queries in (Faloutsos, 1988), and (Jagadish, 1990), respectively. Given a range query, the number of continuous runs on Gray, Z-order, and Hilbert are evaluated in (Jagadish, 1990). Hilbert curve is used as a spatial access method in (Faloutsos & Rong, 1991; Faloutsos & Roseman, 1989), where the data is stored in a one-dimensional disk based on Hilbert values. Hilbert curve is also used as multi-dimensional indexing method (Lawder & King, 2000) and spatio-temporal indexing methods (Jensen et al, 2004). Recently, concurrent spatial indexing methods based on Hilbert curve are proposed in (Dai & Lu, 2007, 2009). More importantly, those experiments show that space-filling curve approach is more preferable in high dimensional space than the R-tree family. Previous works demonstrated that among these SFCs, Hilbert curve has the optimal clustering property (Abel & Mark, 1990; Jagadish, 1990; Mokbel et al, 2003) over a variety of computing conditions. In other words, Hilbert curve provides the best linear mapping to preserve the locality between multi-dimensional objects in one-dimensional space. Given that the data objects are physically arranged on disk according to their SFC values, Hilbert curve is more likely to organize the spatially adjacent data into the same disk page or consecutive disk pages, and hence reduces the required disk accesses for a spatial query. Nevertheless, even using Hilbert curves, a range query may cover several discrete clusters (set of cells with consecutive SFC values), which leads to multiple index tree traversals. The reason is that linear mapping loses spatial relationship between spatial objects. Although objects near to each other in a linear ordering must also be close in the spatial space, the opposite is not always true. In some cases, two neighbors in a spatial space may be far away from each other in the one-dimensional ordering. Recently, Partitioned Hilbert curve has been used to reduce the search range for spatial queries including range and kNN queries in (Zheng et al, 2004). However, the number of clusters covered by a query window is not reduced, and some cells outside the query window may still be in the search range. In order to improve the query response time by reducing the number of clusters (or continuous runs), approximate NN query based on multiple shifted Hilbert curves is proposed by (Liao et al, 2001).

In this chapter, an efficient spatial range query method is designed for compensating the lost of spatial relationship by the linear mapping mechanisms. Hilbert space-filling curve is chosen to map spatial space into the one-dimensional domain, because of its best clustering property. Different from previous work, the proposed method uses multiple copies of Hilbert curves, and each has different rotations or shift. When a range query comes, the Hilbert curve that generates the minimum number of clusters will be selected. Theoretical proofs are provided to show that the rotation of Hilbert curve can reduce the number of clusters significantly. The experiments conducted on real data sets demonstrate that the proposed approach is efficient and scalable, and the combination of rotation and shift outperforms applying any of them independently.

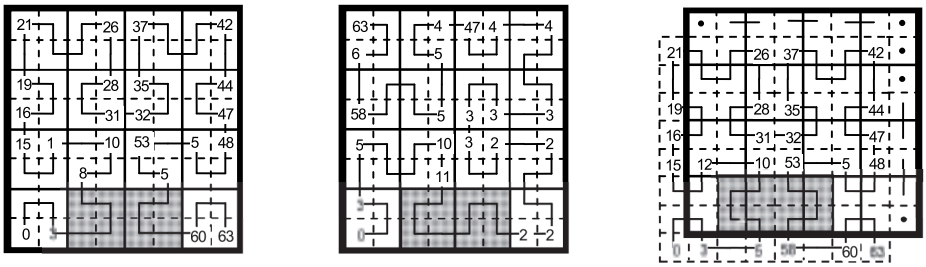
## 2. Motivation

A range query searches all objects that overlap with a given region (also called query window). When Hilbert curve is used as indices, an intuitive approach to answer a range query is to search on the Hilbert values of the cells that overlap with the query window. This procedure consists of three steps, mapping, filtering, and refinement. First, the query window is mapped to a set of cell numbers according to the Hilbert curve traversal. The cells connected consecutively by the curve forms a cluster that indicates a single continuous query. Second, for each cluster, the ranges of cell numbers, i.e., Hilbert values are used to query the corresponding B+-tree. All leaf entries of the B+-tree with their key values exactly the same as the cell numbers within the query window are identified. These entries point to the disk pages that store the objects that potentially overlap with the query window. Third, each object retrieved from those disk pages will be validated to make sure that the object actually overlaps with the query window.

To map multi-dimensional points into one-dimensional values using Hilbert space-filling curve, there is an existing algorithm with the time complexity of $O(kn)$ (Lawder et al., 2001), where $k$ is the order of the curve and $n$ is the dimensionality of the data space. By contrast with the light CPU computation in the mapping step, I/O cost for index traversals in the filtering step is a dominating factor of the time cost of a spatial range query. The I/O cost of the filtering step depends on the number of clusters covered by the range query. *Fig. 1*(a) gives an example of a range query in a two-dimensional space. The data space is divided by 64 uniform grid cells ordered according to Hilbert values. The shaded area represents a range query *A*. *A* overlaps with 8 cells (4,5,6,7,56,57,58,59) that consist of 2 clusters (4-7), (56-59). Using traditional methods, it is necessary to traverse the B+-tree at least 2 times to find all candidate cells that overlap with A. The underlying reason is that some spatial relationships between spatial objects are lost by applying linear mapping. In this example, the two clusters are adjacent in the two dimensional space, but far away from each other in the Hilbert ordering. To reduce the tree traversal times, one intuitive solution is to expand two clusters to form a big cluster, i.e., (4-59). However, a problem of this solution is that it will cause many unnecessary data accesses, i.e., data pages between 8 and 55. Another solution could be enlarging the cell size, but the overhead of accessing additional data space and filtering will be increased accordingly. The following sections focus on reducing the number of clusters of each range query, and meanwhile remaining the refinement overhead. It is observed that the number of clusters covered by a query window varies under Hilbert curves with different orientations and shift. The following two subsections describe the observations in detail.

### 2.1 Rotation

A range query covers different number of clusters when using Hilbert curves with different orientations. Hilbert curve is a recursive space-filling curve. Specifically, in two-dimensional space, the $i^{th}$-order curve is derived by replacing each quadrant with the $(i-1)^{th}$-order curve, and two of them are rotated 90 degree clockwise and anticlockwise respectively. A Hilbert curve in a two-dimensional space has four orientations. The same query window on differently oriented Hilbert curves may derive different number of clusters. Fig. 1(a) and (b) show an example. Hilbert curves with two orientations are applied on the same data space. For the same range query *A*, the two curves result in different numbers of clusters: two in (a) and only one in (b). Note that this change on the number of clusters within a given query window may vary with different query positions.



(a) Original (b) Rotated (c) Shifted

Fig. 1. Range Query A.

To study the relations between the orientations of a Hilbert curve and the number of clusters covered by a query window, we start from the simplest case which is described in Fig. 2. The same notations in (Moon et al., 2001) for the orientation of Hilbert curves are used here. In Fig. 2, a 16-cell grid space is shown in the leftmost. Assuming the size of the query window is 2 * 2, the total number of different query positions is 9, indicated by the numbers shown on the curve in the figure. Fig. 2 (a), (b), (c), and (d) use a 3 * 3 matrix to present the number of clusters contained in the query window, on 2+-orientation, 1+-orientation, 2--orientation, and 1--orientation, respectively. In this matrix, each entry corresponds to a query position, such as the top left entry stands for the position 1. As can be seen, when the query window is located in the top row of the space, the orientation represented in 2(a) gives the fewest clusters. However, when the query window is located in the bottom row of the space, the orientation represented in 2(c) gives the fewest clusters. Based on the matrices of the four orientations, the minimum number of clusters for each position can be calculated, as illustrated in the final matrix 2(e). In 2(e), all positions except the center position have only one cluster. Based on this observation, a fact can be found is that using Hilbert curves with different orientations may reduce the number of clusters for a query window which are not located in the center position.

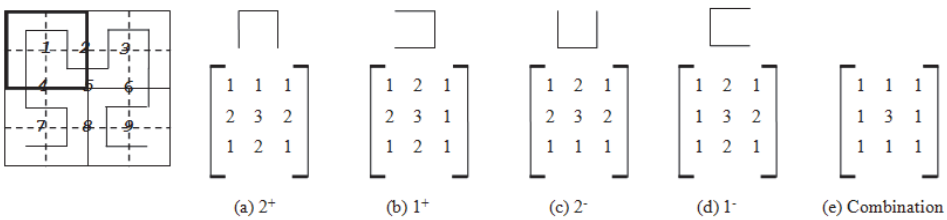
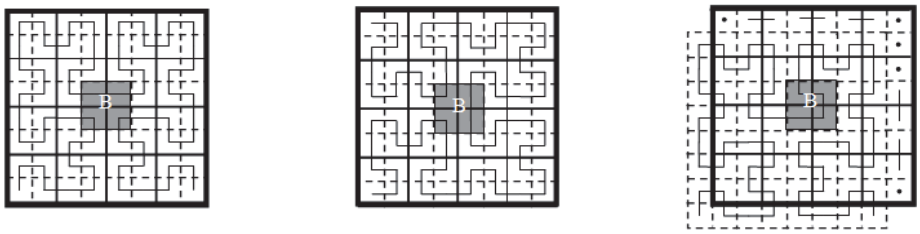



Fig. 2.Clusters with different orders.

### 2.2 Shift

The number of clusters may be decreased when the Hilbert curve is shifted diagonally. The example presented in Fig. 2 shows that when the query window is located at the center of the grid space, the number of clusters can not be reduced with multiple orientations. A similar example can be found in Fig. 3 (a) and (b), where the number of clusters in query window B remains unchanged after the rotation. The reason is that in a two-dimensional case, whenever the order of a Hilbert curve increases, it splits the whole space quarterly,

replaces each sub-space using the original or rotated Hilbert curves, and uses three connection edges to link the four quadrants. Among the three connection edges, only one is contributed to the center of the entire space, while the other two are for boundaries. However, when the Hilbert curve is diagonally shifted, as shown in Fig. 3 (c), the same query window will cover only one cluster.




(a) Original (b) Rotated (c) Shifted

Fig. 3. Range query B.

### 2.3 Hybrid

| Symbols | Definition |
|---|---|
| $H_{k+n}$ | 2-dimensional Hilbert curve with size $2^{k+n} * 2^{k+n}$ |
| $H_k$ | 2-dimensional Hilbert curve with size $2^k * 2^k$ |
| $t_n$ | Number of connection edges within the top boundary of a $2^+$-*oriented* $H_{k+n}$. |
| $b_n$ | Number of connection edges within the bottom boundary of a $2^+$-*oriented* $H_{k+n}$. |
| $s_n$ | Number of connection edges within one side boundary of a $2^+$-*oriented* $H_{k+n}$. |
| $c_{t,n}$ | Number of connection edges between $H_k$ sub regions in top boundary and in other areas of a $2^+$-*oriented* $H_{k+n}$. |
| $c_{b,n}$ | Number of connection edges between $H_k$ sub regions in bottom boundary and in other areas of a $2^+$-*oriented* $H_{k+n}$. |
| $B_{i+/-,n}$ | Number of $i^{+/-}$-*oriented* $H_k$ in the bottom boundary of a $2^+$-*oriented* $H_{k+n}$. |
| $T_{i+/-,n}$ | Number of $i^{+/-}$-*oriented* $H_k$ in the top boundary of a $2^+$-*oriented* $H_{k+n}$. |
| $h_k$ | Number of horizontal edges in a 2-*oriented* $H_k$. |
| $v_k$ | Number of vertical edges in a 2-*oriented* $H_k$. |
| $N_{i+/-,t}$ | Average number of clusters within $2^k * 2^k$ query region in the top boundary of $2^{k+n} * 2^{k+n}$ $i^{+/-}$-*oriented* $H_{k+n}$. |
| $N_{i+/-,b}$ | Average number of clusters within $2^k * 2^k$ query region in the bottom boundary of $2^{k+n} * 2^{k+n}$ $i^{+/-}$-*oriented* $H_{k+n}$. |

Table 1. Notations.

From the above observations, a range query can have different numbers of clusters under Hilbert curves with different orientations or shifts. Thereby, using multiple copies of rotated or shifted Hilbert curves should reduce the number of clusters for range queries in general. However, if only rotation is used, there is a "center position" problem. As described previously, shift can reduce the number of clusters when the query window is located in center position. However, it is not always better to shift for other range queries. For example, in Fig. 1 (c), the shift result of (a), still has the same number of clusters for range query A. It can be concluded from these observations that, in some cases, rotation is better

than shift, but some cases may be opposite. Therefore, combining the rotation and shift can be more effective than applying any single one of them independently.

## 3. Spatial range query algorithms

In this section, we provide a theoretical analysis on the first observation, and then derive a formula to measure the improvement of applying multiple copies of Hilbert curves with different orientations. We also introduce a new spatial range query algorithm designed based on the combination of rotations and shift.

### 3.1 Theoretical proofs

In this section, formulas will be derived to calculate the average number of clusters for a given query region in the top and bottom boundary of a 2+-oriented Hilbert curve. And then we prove that the average number of clusters within given query region on 2--oriented Hilbert curve is smaller than the average number of clusters on 2+-oriented Hilbert curve, when the queries are located on the bottom boundary of the space. This proof can be extended to queries located in other areas and Hilbert curves with other orientations. Specifically, we assume that the query window is a region with size $2^k * 2^k$, and the size of the grid space is $2^{k+n} * 2^{k+n}$. The notations used in the proof are listed in Table 1. We define connection edge in a $2^{k+n} * 2^{k+n}$ Hilbert curve as the edge that connects two sub curves, each with size $2^k * 2^k$.

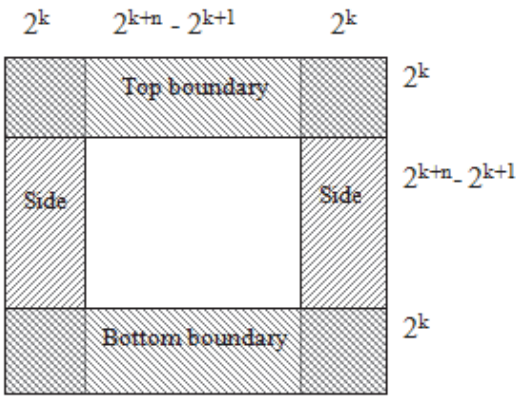



Fig. 4. $H_{k+n}$ divided into 9 subregions.

The grid space of $H_{k+n}$ is divided into nine sub regions, as shown in *Fig. 4.* The smaller side length of each sub region on the boundary is 2k. Then, the $2^{k+n} * 2^{k+n}$ grid region $H_{k+n}$ can be considered as a collection of $2^{2n}$ $H_k$, each of which connects to one or two neighbors by connection edges. The following proves are deducted from parts of the conclusions in (Moon et al., 2001).

By definition of Hilbert curve, 2+-oriented Hilbert curve and 2--oriented Hilbert curve are symmetrical, when given the curve-space, order, so for given query region, the average number of clusters in the top boundary of a 2+-oriented Hilbert curve is equal to the average number of clusters in the bottom boundary of a 2--oriented.

**Remark 1**: *The difference of the average number of clusters between 2+-oriented Hilbert curve and 2--oriented Hilbert curve when queries are located in the bottom boundary of the curve-space is equal to the difference between those of the bottom boundary and the top boundary of 2+-oriented Hilbert curve, for the same query region.*

From (Moon et al., 2001), we have 1) which gives formula to calculate the number of connection edges in the top boundary, and the relationship between the number of connection edges in the bottom boundary and those in the side boundary; 2) which states there is only 2+-oriented $H_k$ on the top boundary of 2+-oriented $H_{k+n}$, and no 2--oriented $H_k$ on the bottom boundary of 2+-oriented $H_{k+n}$; 3) which presents the relationship between the numbers of differently oriented $H_k$ in the bottom boundary of 2+-oriented $H_{k+n}$. Based on this, the formulas to calculate the exact number of connection edges in bottom and side boundary, and the number of $H_k$ in the bottom boundary are derived in the following Lemma 1 and Lemma 2, respectively.

**Lemma 1:** For any positive integer n,

$$b_n = (2^{n+1} + (-1)^n)/3 - 1, s_n = (2^{n+2} - 3 - (-1)^n)/6.$$

**Proof**.

$$\begin{aligned} & s_n = s_{n-1} + 2s_{n-2} + 1 \\ & \Rightarrow\ s_n + s_{n-1} = 2(s_{n-1} + s_{n-2}) + 1 \Rightarrow\ s_n + s_{n-1} = 2^n - 1 \\ & \Rightarrow s_n = (2^{n+2} - 3 - (-1)^n)/6. \end{aligned}$$

**Lemma 2:**

$$B_{2^+,n} = \frac{2^n + (-1)^n 2}{3}, B_{1^+,n} = B_{1^-,n} = \frac{2^n - (-1)^n}{3}.$$

These can be proved in the similar way as Lemma 1.

So far, the number of connection edges and the number of $H_k$ inside the top or bottom boundary are derived. Next, the number of connection edges connecting the top or bottom boundary to the other areas need to be obtained.

**Lemma 3:**

$$c_{t,n} = 2^n, \quad c_{b,n} = (2^{n+1} - 2(-1)^n)/3.$$

**Proof.**

There are only *2+-oriented* $H_k$ in the top boundary of a *2+-oriented* $H_{k+n}$. Each of them has two end points (one incoming point and one outgoing point). One end point connects to the adjacent *2+-oriented* $H_k$ in the top boundary and another connects to a sub curve inside boundaries or center area. Accordingly, $c_{t,n}$ is equal to the number of *2+-oriented* $H_k$ in the top boundary, i.e., $2^n$.

Similarly, $c_{b,n}$ is equal to the sum of the numbers of *1+-oriented* and *1--oriented* $H_k$ in the bottom boundary of a *2+-oriented* $H_{k+n}$, because the *2+-oriented* $H_k$ does not contribute to connections to the other areas and there is no *2--oriented* $H_k$ in the bottom boundary.

It is known that the number of clusters within a query region is equal to half the number of edges cut by the boundary of the region. Each connection edge in the top and bottom

boundary is horizontal and cut twice by the left and right sides of query windows; each horizontal edge in a $H_k$ of the top or bottom boundary is also cut twice by the left and right sides of query windows; each edge connecting the top and bottom boundary to the center area is vertical and is cut $2^k$ times by the top or bottom sides of query windows, except those edges in the two side boundary, which is cut once only.

As defined in *Table 1*, $h_k$ and $v_k$ denote the number of horizontal and vertical edges in a *2-oriented* $H_k$, so they indicate the vertical and horizontal edges in a *1-oriented* $H_k$, respectively.

In the top boundary of the $H_{k+n}$, the total number of the possible positions of the query window $2^k * 2^k$ is $2^{k+n}-2^k+1$. Therefore, we derive the formula for calculating the average number of clusters of the query window located on the top/bottom boundary of *2+-oriented* Hilbert curve as follows.

**Theorem 1:** *The average number of clusters of a* $2^k * 2^k$ *query window located in the top boundary and bottom boundary of a* $2^{k+n} * 2^{k+n}$ *grid space which is 2+-oriented* $H_{k+n}$ *are equal to*

$$N_t = \frac{2(T_{2+,n} * h_k + t_n + 1) + 2^k(c_{t,n} - 2)}{2(2^{k+n} - 2^k + 1)} = \frac{2^{k+n} + 2^{n+1} * h_k + 2^n + 2 - 2^{k+1}}{2(2^{k+n} - 2^k + 1)}$$

$$N_b = \frac{2(B_{2+,n} * h_k + (B_{1+,n} + B_{1-,n}) * v_k + b_n) + 2^k(c_{b,n} - 2) + 2}{2(2^{k+n} - 2^k + 1)}$$

$$= (\frac{2^n + 2(-1)^n}{3} h_k + \frac{2^{n+1} - 2(-1)^n}{3} v_k$$

$$+ \frac{2^{n+1} + (-1)^n}{3} + \frac{2^{k+n} - 2^k(-1)^n}{3} - 2^{k+1}) / (2^{k+n} - 2^k + 1).$$

**Note.** For a $2^+$/-oriented $H_k$, the number of vertical edges is one more than the number of horizontal edges by definition.

**Corollary 1:** *The difference between the average number of clusters on top boundary and bottom boundary for a 2+-oriented* $H_{k+n}$ *can be derived*:

$$N_b - N_t = \begin{cases} \frac{2^{n+1} + 2^{n-1} - 2 + h_k + 2^{k+n-1} - 2^{k+1}}{3(2^{k+n} - 2^k + 1)}, & n \text{ is even}, \\ \frac{2^{n+1} + 2^{n-1} - 3 + h_k + 2^{k+n-1} + 2^{k+1}}{3(2^{k+n} - 2^k + 1)}, & n \text{ is odd}. \end{cases}$$

The number of clusters for the side boundary can be derived with the similar idea. Although the above formula expresses the calculation on *2+-oriented* Hilbert curve, it is still applicable to all 2-dimensional Hilbert curves with other orientations. From the above theorem, we note that the top boundary of the *2+-orientation*, the bottom boundary of the *2--orientation*, the right side boundary of the *1+-orientation*, and the left side boundary of the *1--orientation* contains the fewest clusters for a given query window size comparing with the any other orientations at the same position.

### 3.2 Algorithms

#### 3.2.1 Index construction

According to the first observation, we create four B+-trees for the same data set based on the four Hilbert curves with different orientations. These curves have identical curve-space, order, and the cell size (granularity). The B+-trees and corresponding Hilbert curves are

**Algorithm for spatial range query**
**Procedure** **RANGEQUERY** (*qw*)
**Input:** *qw:* query window
**Output:** *RS*: set of objects which overlap with *qw*.
1. *ClusterList[]* ← empty

//Mapping query window into clusters based on every Hilbert curve, ($H_0$: *Origin*, $H_1$: *Right*, $H_2$: *Left*, $H_3$: *Down and* $H_4$: *Shift*).
2. **for** $i = 0$ to 4
3. *Cells[]*← Hilbert codes of cells of $H_i$ overlap with *qw*
4. *Clusters[][]*← group cells into clusters
5. *ClusterList[i]* ← <$H_i$, *Clusters*>
6. **end for**

//Select the Hilbert curve with the smallest numbers of clusters, and load the corresponding B+-tree.
7. Find *k*, such that *ClusterList[k]* = Min(*ClusterList[0]*... *ClusterList[n]*)
8. *T* ← load B+-tree corresponding to $H_k$

//Traverse B+-tree, find out candidate data objects.
9. **for** each cluster $C_j$ in *ClusterList[k].Clusters*
10. Traverse tree *T*, Get data page *p*,
11. *Objs[]* ← every objects stored in *p*
12. **end for**

//Refinement
13. **for** every object in *Objs[]*
14. **if** *Objs[i]* overlaps with *qw* then
15. *RS* ← *Objs[i]*
16. **end if**
17. **end for**

18. **return** *RS*

Fig. 5. Range query algorithm.

named in terms of the orientation of the Hilbert curves. Specifically, the $2^+$*-oriented* Hilbert curve and the corresponding B+-tree are named as "Origin", the $2^-$*-oriented* Hilbert curve, and the corresponding tree are named as "Down", similarly, the $1^+$*-oriented* as "Right" and the $1^-$*-oriented* as "Left". According to the second observation, another B+-tree, "Shift", is also created for the same data set. For instance, if the original data space is of the range [0, 1] on each dimension, the shifted range will be [s, 1+s] on all dimensions respectively, where *s* is the side length of a cell. To calculate the cells located in the area $[1, 1+s]^d$, (*d* represents dimension), the Hilbert curve space needs to be enlarged to [0, 2] on each dimension, meanwhile the order will be increased by 1. Therefore, "Shift" is generated using the same cell size as the original Hilbert curve, and doubled curve space. In "Shift", each data point is shifted up-right by one cell. For example, a point in original data space is $p(x, y)$, it will be changed to $p'(x+s, y+s)$ before calculating the Hilbert value, and then be inserted into "Shift" with the new Hilbert value as the key. Although multiple indices are created for one data set, the data objects are stored in disk based on their "Origin" Hilbert curve values. Reasonably, we assume that there is a page buffer to reduce additional data page seek time by sorting the addresses of data pages before accessing them physically.

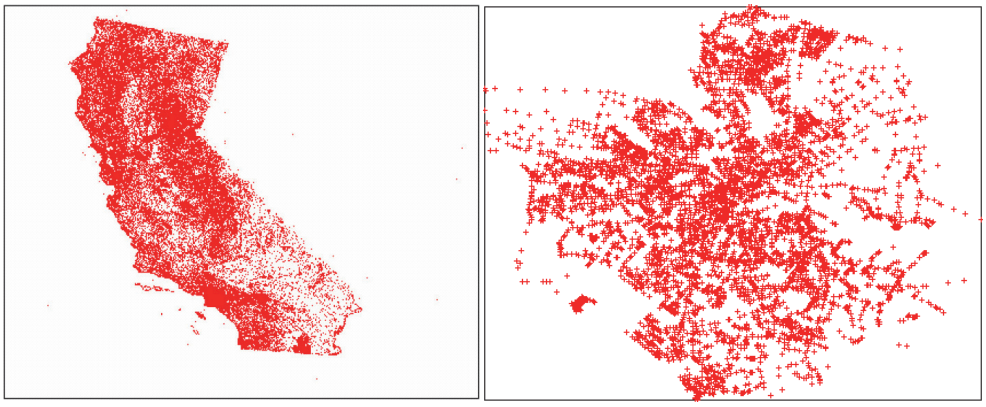
### 3.2.2 Mapping and filtering

The detailed algorithm for processing range query based on multiple copies of Hilbert curves is presented in Fig. 5. The example shown in Fig. 1 can be used to illustrate this algorithm. In this example, the whole data space is [0, 8] * [0, 8]; the cell size is 1; the order of the curve is 3; and the query window A is < (2, 0), (6, 2)>. The clusters covered by *A* on the five Hilbert curves are calculated at first. To compute the clusters under shifted Hilbert curve, the region of the query window needs to be recalculated, since the whole data space has been shifted. For example, the query window *A*< (2, 0), (6, 2)> is transformed to *A'*< (3, 1), (7, 3)>. A data structure *ClusterList* is used to store the cluster information. Each entry of the list represents clusters for one Hilbert curve, in the form of <Curve name, [cluster1]… [clusterN]>. In this example, the *ClusterList* contains three entries, <"Origin", ([4-7][56-59])>, <"Right", ([12-19])>, and <"Shift", ([6-9][54-57])>. The index corresponding to the entry with fewest clusters is selected, e.g., the index "Right" is used for answering range query *A*. In case that more than one Hilbert curves produce the fewest clusters, the one that has smaller sum of gaps between clusters will be selected. Because when the gap between two clusters is small, the corresponding leaf nodes of the second cluster can be located quickly from the first cluster, by just following links between leaf nodes.

### 3.2.3 Refinement

After the data objects are obtained from the filtering step, further validation is needed to check the overlaps between query window and these retrieved objects. If an object overlaps with the query window, it will be put into the result set. Otherwise, the object will be removed. This step is similar to the refinement of the traditional spatial range query processing approach.

## 4. Experiment



(a) California Places (b) City of Oldenburg

Fig. 6. Datasets.

To demonstrate the efficiency of the proposed algorithm and the correctness of the analysis, we conducted experiments to evaluate the performance of range queries by comparing with access method using only one Hilbert curve. The I/O costs of range queries with various sizes and positions are examined on the proposed method with different combinations of rotations and shift. The objective of our experiments is to assess the efficiency of different combinations of rotations and shift.

### 4.1 Experiment design

The experiment is performed on point data sets downloaded from (Sequoia 2000) and collection of real road network (R-tree portal; Li et al., 2005). The two data sets are shown in Fig. 6. The one from Sequoia 2000 is composed of more than 62 thousand 2-dimensional points, which represents places in California; another, from a collection of road network, presents about 6,000 road network's nodes in the city of Oldenburg. The experiments are conducted as illustrated in Fig. 7. The average number of page access for several range queries with difference size are compared based on the different copies of Hilbert curves. The size of the query window ranges from 1% to 15% of the whole data space. To obtain exact measurements of the average number of clusters, all possible positions for different range query sizes are examined over the whole grid space. Multiple B+-trees are constructed based on Hilbert codes of data points computed from Hilbert curves with variant orientation and shift. We compared the performance achieved by multiple Hilbert curves to that of the original approach, which uses only one Hilbert curve, as well as the performance of different combinations of rotation and shift. The performance is measured by the average number of page accesses in the B+-tree for a range query.

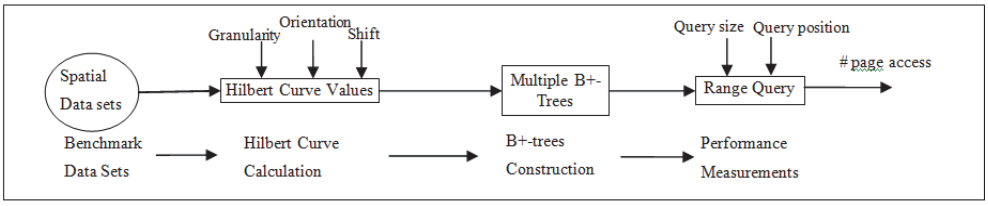



Fig. 7. Experimental design.

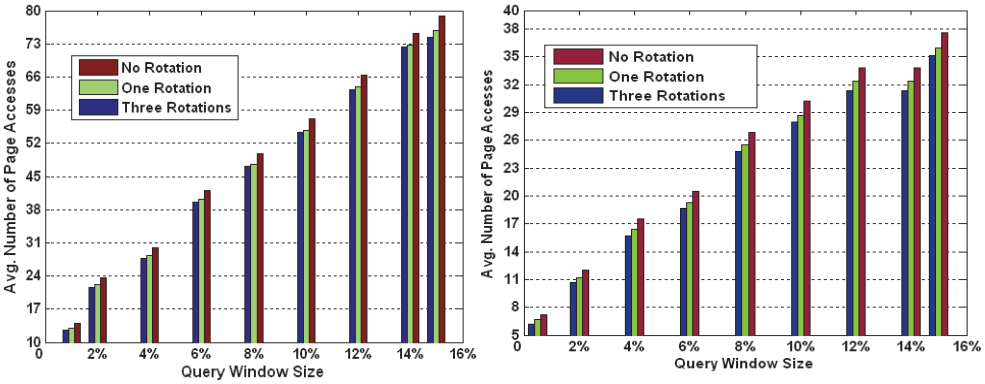
### 4.2 Experiment results

#### 4.2.2 Effect of different number of rotations

Fig. 8 shows the comparison of range queries on different numbers of rotations. The query window size varies from 1% to 15% over the whole data space for both data sets. As shown in both Fig. 8 (a) and (b), the average number of page accesses increases with the growth of the query size. Consistent to theoretical analysis, when multiple Hilbert curves with variant orientations are used, the average number of page accesses is less than that of only one Hilbert curve. Moreover, the I/O cost saved by applying multiple Hilbert curves is enhanced with the increase of the query size. Observed from the results, using four orientations definitely reduces more I/O cost than two orientations. However, the performance gained by using two orientations from one orientation is more remarkable than the performance improved by using four orientations from two orientations. Based on this

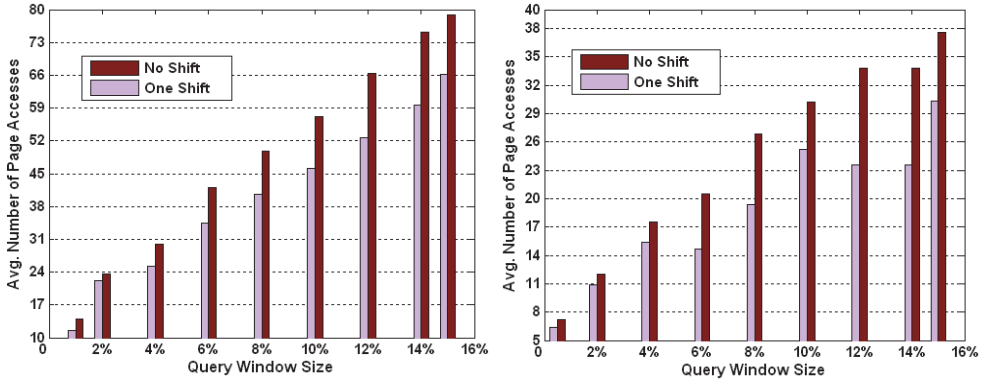
conclusion, there is a tradeoff between the performances improvement by using multiple Hilbert curves and the storage space required to store additional copies of indices. It depends on different applications to determine how many orientations are most appropriate. For space sensitive applications, two orientations may be deployed rather than using all four orientations, considering the additional space requirement. However, for the applications in which query efficiency is most crucial, applying all four orientations may be a better choice.

### 4.2.2 Effect of shift




(a) California Places (b) City of Oldenburg

Fig. 8. Comparison of different rotations.




(a) California Places (b) City of Oldenburg

Fig. 9. Comparison on shift.

Fig. 9 describes the effect of using one additional copy of the Hilbert curve with shift. We set the same parameter values such as query size, position, order of Hilbert curve, as the first experiment. The average number of page accesses is significantly reduced by using shift comparing to using only one Hilbert curve. For instance, when the query size is over 12%,

the average number of page accesses is reduced up to 30%. As a similar trend observed here as the effect of rotations, with the query size increasing, the number of reduced page accesses by shift also increases. However, the average number of page accesses of different query size presents a zigzag form in the case of using shift technique on the second data set. It is observed that the average number of page accesses decrease when the size of a query window happens to consist of integral number of 2*2 cell-blocks. For example, in Figure 9(b), when the size of query window is 6% of the whole space, the side length of the query window is 8, so that it contains 16 2*2 cell-blocks. The reason is that when the query range size meets the above condition, cells covered by the query window tend to be grouped in the same cluster along the Hilbert curve, by choosing shifted or original space. The shift technique can increase the probability that a range query contains only one cluster, even if it has several clusters on the original Hilbert curve. While in case of multiple rotations, if the range query contains multiple clusters on the original Hilbert curve, it can not consist only one cluster with any rotations. Fig. 3 is an example. The size of the query B is 2*2, and it contains 3 clusters with any rotations, but only one cluster on the shift.

#### 4.2.2 Effect of hybrid

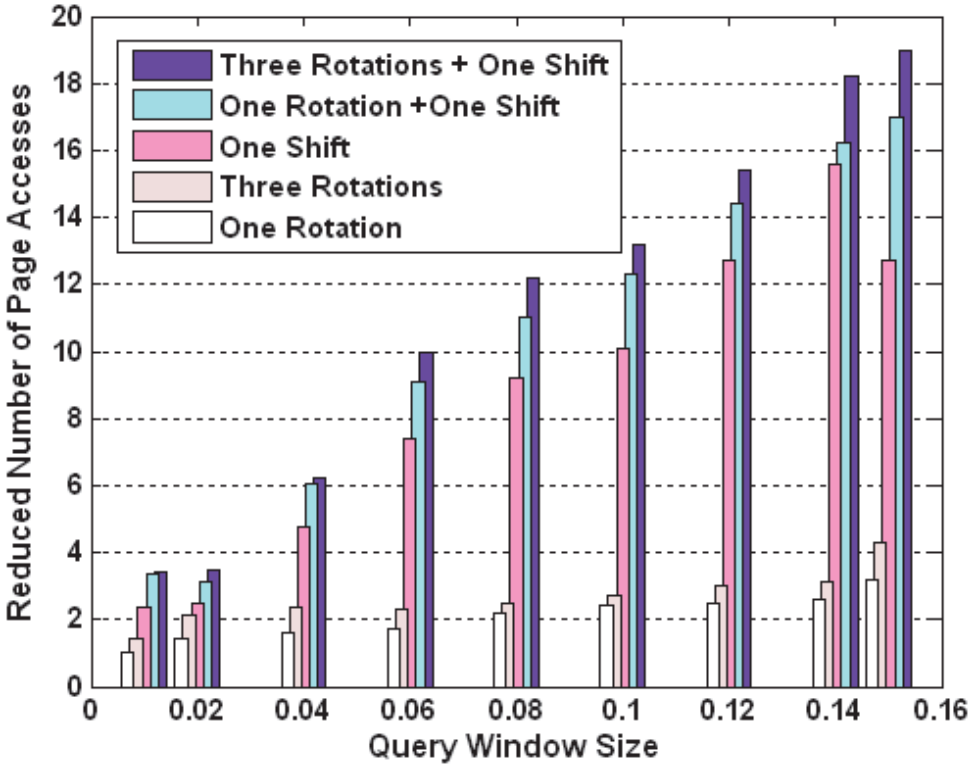



Fig. 10. Efficiency comparisons.

Fig. 10 illustrates the comparisons between different combinations of rotations and shift, rotations only and shift only. Comparisons are based on the number of page accesses reduced comparing to the original approach on California Places. As can be observed from the figure, the different combinations can be ordered by the number of page accesses as follows: One Rotation < Three Rotations < Shift < One Rotation + Shift < Three Rotations + Shift. Along this ordered sequence of the combinations, the gap between Shift and Three Rotations are the largest. This indicates that rotations do not reduce I/O cost as significantly as shift does. However, combining all rotations and shift performs better than applying any one of them independently, consistent to what we deduced in Section 2.3.

## 5. Conclusions

This chapter proposes an efficient spatial range query processing method based on rotation and shift techniques. Facts are observed that the same query on Hilbert curve with different orientations and shift obtains different numbers of clusters. Theoretical analysis is also provided to prove that multiple copies of Hilbert curves with different orientations can reduce the number of clusters of a range query. The experiments on two real data sets demonstrate that the proposed method reduces I/O costs of range queries. The results show that the combinations of rotation and shift in general provide the better performance than applying any one of them independently.

Future directions from this work include: investigation on jumps between clusters to further improve query performance, theoretical analysis on the effectiveness of shift, and designing spatial operations such as KNN, spatial join, and moving object queries utilizing multiple Hilbert curves.

## 6. References

Abel, D. J. & Mark, D. M. (1990). A Comparative Analysis of Some Two-Dimensional Orderings, International J. Geographical Information Systems, Vol. 4, No. 1, pp. 21-31.

Dai, J. & Lu, C.-T. (2007). CLAM: Concurrent Location Management for Moving Objects, Proceedings of the 15th ACM International Symposium on Advances in Geographic Information Systems (ACMGIS), pp. 292-299.

Dai, J. & Lu, C.-T. (2009). A Concurrency Control Protocol for Continuously Monitoring Moving Objects, Proceedings of the 10th International Conference on Mobile Data Management (MDM), pp. 132-141.

Faloutsos, C. (1988). Gray Codes for Partial Match and Range Queries, IEEE Transactions on Software Engineering, Vol. 14, No. 10, pp. 1381-1393.

Faloutsos, C. & Rong, Y. (1991). A Spatial Access Method Using Fractals, Proceedings of the International Conference on Data Engineering (ICDE), pp. 152-159.

Faloutsos, C. & Roseman, S. (1989). Fractals for Secondary Key Retrieval, Proceedings of the 8th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), ACM Press, New York, NY, pp. 247-252.

Gray, F. (1953). Pulse Code Communications, US Patent 2632058, 1953. Hightower, J., Vakili, C., Borriello, C., & Want, R. (2001). Design and calibration of the SpotON

ad-hoc location sensing system, University of Washington Technical Report CSE 00-02-02.
http://www.cs.washington.edu/homes/jeffro/pubs/hightower2001design/hightower2001design.pdf

Hilbert, D. (1891). Ueber stetige abbildung einer linie auf ein flashenstuck, Mathematishe annalen, pp. 459-460.

Jagadish, H. V. (1990). Linear Clustering of Objects with Multiple Attributes, Proceedings of the ACM SIGMOD Conference on Management of Data, ACM Press, New York, NY, pp. 332 - 342.

Jensen, C. S., Lin, D., Ooi B.C. (2004). Query and Update Efficient B+-Tree Based Indexing of Moving Objects, Proceedings of the 30th International Conference on Very Large Data Bases (VLDB), pp. 768-779.

Lawder, J. K. & King, P. J. H. (2000). Using Sapce-filling Curves for Multi-dimensional Indexing, Proceedings of the 17th British National Conference on Databases (BNOD), pp. 20-35.

Lawder, J. K. & King, P. J. H. (2001). Using State Diagrams for Hilbert Curve Mappings, International Journal of Computer Mathematics, Vol. 78, No. 3, pp. 327-342.

Li, F., Cheng, D., Hadjieleftheriou, M., Kollios, G., and Teng, S.-H. (2005). On Trip Planning Queries in Spatial Databases, Proceedings of the 9th International Symposium on Spatial and Temporal Databases (SSTD).

Liao, S., Lopez, M. A., & Leutenegger, S. (2001) High Dimensional Similarity Search with Space-Filling Curves, Proceedings of the International Conference on Data Engineering (ICDE), pp. 615-622.

Mokbel, M. F., Aref, W. G., & Kamel, I. (2003) Analysis of Multi-dimensional Space-Filling Curves, GeoInformatica, Vol. 7, No. 3, pp. 179-209.

Moon, B., Jagadish, H. V., Faloutsos, C., & Saltz, J. H. (2001). Analysis of the Clustering Properties of the Hilbert Space-Filling Curve, IEEE Transactions on Knowledge and Data Engineering, Vol. 13, No. 1, pp. 124-141.

Ni, L. M., Liu, Y., Lau, Y. C., & Patil, A. P. (2004). LANDMARC: Indoor Location Sensing Using Active RFID, Wireless Networks, Vol. 10, pp. 701–710.

Orenstein, J. A. & Merrett, T. H. (1984). A Class of Data Structures for Associative Searching, Proceedings of the 3rd ACM SIGACT-SIGMOD-SIGART Sympsium on Principles of Database System (PDOS), ACM Press, New York, NY, pp. 326-336.

Peano, G. (1890). sur une courbe qui remplit toute une air plaine, Mathematishe Annalen, Vol. 36, pp. 157-160.

Simmons, G. F. (1963). Introduction to Topology and Modern Analysis, New York, McGraw-Hill Book Company, 1963.

Willis, S. & Helal, S. (2004). A Passive RFID Information Grid for Location and Proximity Sensing for the Blind User, University of Florida Technical Report, TR04-009.
http://www.cise.ufl.edu/tech_reports/tr04/tr04-009.pdf

Zheng, B., Lee, W.-C., & Lee, D. L. (2004). Spatial Queries in Wireless Broadcast Systems, Wireless Networks, Vol. 10, No. 6, pp. 723-736.
R-tree portal: http://www.rtreeportal.org/.
Sequoia 2000: http://s2k-ftp.cs.berkeley.edu:8000/sequoia/.