2002

Performance of Multi-Dimensional Space- Filling Curves

Mohamed F. Mokbel

Walid G. Aref
Purdue University, aref@cs.purdue.edu

Ibrahim Kamel

Report Number:
02-028

PERFORMANCE OF MULTI-DIMENSIONAL SPACE-FILLING CURVES

Mohamed F. Mokbel
Walid G. Aref
Ibrahim Kamel

CSD TR #02-028
December 2002

Performance of Multi-Dimensional Space-filling Curves

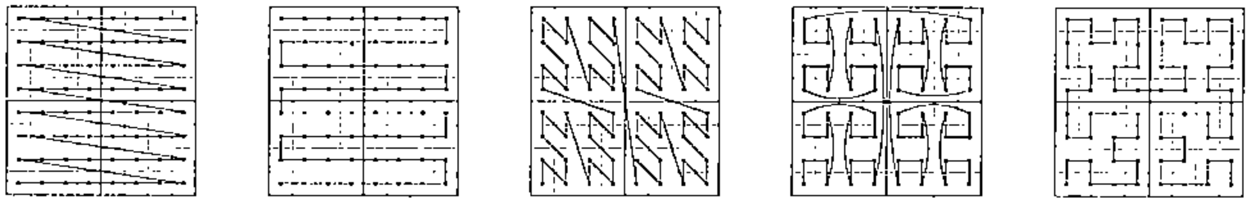
Mohamed F. Mokbel[1] Walid G. Aref[1] Ibrahim Kamel[2]
[1]Department of Computer Sciences, Purdue University, West Lafayette, IN 47907-1398
[2]Panasonic Information and Networking Technologies Laboratory. Two Research Way Princeton, NJ 08540
{mokbel,aref}@cs.purdue.edu, ibrahim@research.panasonic.com

Abstract

A space-filling curve is a way of mapping the multi-dimensional space into the one-dimensional space. It acts like a thread that passes through every cell element (or pixel) in the D-dimensional space so that every cell is visited exactly once. There are numerous kinds of space-filling curves. The difference between such curves is in their way of mapping to the one-dimensional space. Selecting the appropriate curve for any application requires knowledge of the mapping scheme provided by each space-filling curve. A space-filling curve consists of a set of segments. Each segment connects two consecutive multi-dimensional points. Five different types of segments are distinguished, namely, *Jump*, *Contiguity*, *Reverse*, *Forward*, and *Still*. A description vector $V = (J, C, R, F, S)$, where J, C, R, F, and S, are the percentages of *Jump*, *Contiguity*, *Reverse*, *Forward*, and *Still* segments in the space-filling curve, encapsulates all the properties of a space-filling curve. The knowledge of V facilitates the process of selecting the appropriate space-filling curve for different applications. Closed formulas are developed to compute the description vector V for any D-dimensional space and grid size N for different space-filling curves. A comparative study of different space-filling curves with respect to the description vector is conducted and results are presented and discussed.

1 Introduction

Mapping the multi-dimensional space into the one-dimensional domain plays an important role in applications that involve multi-dimensional data. Multimedia databases, Geographic Information Systems (GIS), QoS routing and Image processing are examples of multi-dimensional applications. Modules that are commonly used in multi-dimensional applications include searching, sorting, scheduling, spatial access methods, indexing and clustering. Numerous research has been conducted for developing efficient algorithms and data structures for these modules for one-dimensional data. In most cases, modifying the existing one-dimensional algorithms and data structures to deal with multi-dimensional data results in spaghetti-like programs to handle many special cases. The cost of maintaining and developing such code degrades the system performance.



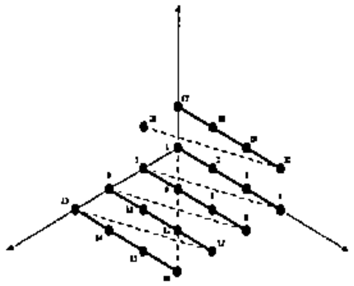
(a) Sweep (b) Scan (c) Peano (d) Gray (e) Hilbert

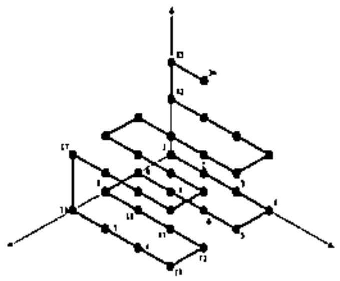
Figure 1: Two-dimensional Space-Filling Curves.

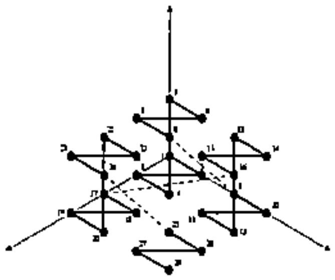
Mapping from the multi-dimensional space into the one-dimensional domain provides a pre-processing step for multi-dimensional applications. The pre-processing step takes the multi-dimensional data as input and outputs the same set of data represented in the one-dimensional domain. The idea is to keep the existing algorithms and data structures independent of the dimensionality of data. The objective of the mapping is to represent a point from the D-dimensional space by a single integer value that reflects the various dimensions of the original space. Such a mapping is called a locality-preserving mapping in the sense that, if two points are near to each other in the D-dimensional space, then they will be near to each other in the one-dimensional space.

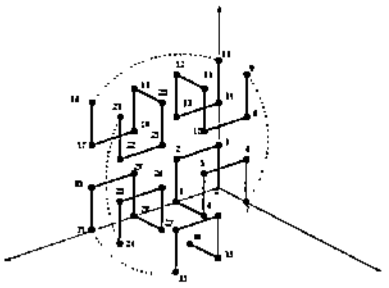
Space-filling Curves (SFCs) have been extensively used as a mapping scheme from the multi-dimensional space into the one-dimensional space. A space-filling curve is a thread that goes through all the points in the space while visiting each point only one time. Thus, a space-filling curve imposes a linear order of points in the multi-dimensional space. Space-filling curves are discovered by Peano [36] where he introduces a mapping from the unit interval to the unit square. Hilbert [20] generalizes the idea to a mapping of the whole space. Following Peano and Hilbert curves, many space-filling curves are proposed, e.g., [6, 30, 39]. Figures 1 and 2 give examples of two- and three-dimensional space-filling curves with grid size (i.e., number of points per dimension) eight and four, respectively. According to the classification in [6], space-filling curves are classified into two categories: recursive space-filling curves (RSFC) and non-recursive space-filling curves. An RSFC is an SFC that can be recursively divided into four square RSFCs of equal size. Examples of RSFCs are the Peano SFC, (Figure 1c), the Gray SFC, (Figure 1d) and the Hilbert SFC, (Figure 1e). For a historical survey and more types of space-filling curves, the reader is referred to [37].

With the variety of space-filling curves and the wide spread of multi-dimensional applications, the selection of the appropriate space-filling curve for a certain application is not a trivial task. One way is to perform many simulation experiments over different space-filling curves. However, this is not practical in terms of execution time. Another way is to tailor a new space-filling curve for each application, e.g., as in [6, 7, 32]. However, with the increase of multi-dimensional applications, it becomes a hard task to tailor a new space-filling curve for each application.


(a) Sweep


(b) Scan


(c) Peano


(d) Gray

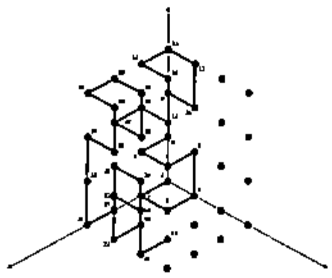

(e) Hilbert

Figure 2: Three-dimensional Space-Filling Curves.

The objective of this paper is to provide a systematic and a scalable framework for selecting the appropriate space-filling curve for any application. To achieve this objective, we divide any space-filling curve into segments. Each segment connects two consecutive multi-dimensional points. Thus, a D-dimensional space-filling curve with grid size N would have N^D-1 segments that connect N^D points. We distinguish among five different segment types *Jump*, *Contiguity*, *Reverse*, *Forward*, and *Still*. A space-filling curve SFC is described by its description vector $V = (J, C, R, F, S)$, where J, C, R, F, and S, are the percentages of *Jump*, *Contiguity*, *Reverse*, *Forward*, and *Still* segments, respectively. Then, with only looking at the description vector V, one can choose the right space-filling curve for a given application.

The rest of this paper is organized as follows. Section 2 surveys some of the related work on space-filling curves. Different types of segments in space-filling curves are presented in Section 3. Section 4 analyzes two non-recursive space-filling curves , the Sweep and Scan SFC, and three recursive space-filing curves, the Peano, Gray and Hilbert SFC, and develops closed formulas to compute the description vector of each space-filling curve. In Section 5, we conduct a comprehensive comparison among different space-filling curves. Finally, Section 6 concludes the paper.

2 Related Work

Although space-filling curves were discovered in the last century [20, 30, 36], their use in computer science applications is not discovered until recently. The use of space-filling curves is motivated by the emergence of multi-dimensional applications. Space-filling curves are used by [33] for spatial join of multi-dimensional data. Multi-dimensional data is transformed into the one-dimensional domain using the Z-order [34], which is the same as the Peano SFC [36]. The transformed data is stored in a one-dimensional data structure, the B^+-Tree [11], and a spatial join algorithm is applied. The Gray [18] and Hilbert [20] SFCs are used for answering range queries in [12, 21], respectively. [14, 15] use space-filling curves as a spatial access method where the multi-dimensional data is stored in one-dimensional media (disk) using the Hilbert SFC. This achieves clustering and hence reduces the number of disk accesses and improves the response time. In [22], the Hilbert SFC is used in packing the R-Tree [19], where a set of rectangles are sorted according to the Hilbert order, and then are packed into the R-Tree nodes. Similar ideas for constructing R-trees using space-filling curves are proposed in [23]. The Z-order [34] (Peano SFC [36]) is used in [9] as a spatial access method to enhance the performance of spatial join. Spatial objects located in a disk are ordered according to their Z-order value to minimize the number of times a given page is retrieved from the disk. Similar use of space-filling curves is performed in [38] based on the Hilbert SFC. The Hilbert SFC is also used in multi-dimensional indexing in [24, 25] and for answering nearest-neighbor queries in [26].

Other uses of space-filling curve include data-parallel applications [35], disk scheduling [4], memory management [27, 40], and image processing [42, 44, 46]. Some applications need a tailored space-filling curve. In [6], a new recursive space-filling curve is proposed that guarantees an upper bound of three seek operations to any two-dimensional square query. In [32], an H-index ordering is proposed for mesh-indexing. XZ-ordering is proposed by [7] to map objects with spatial extension. The XZ-order is an extension of the Z-order by extending each region in Z-order by a factor of two in each dimension.

The properties of different space-filling curves is explored in [3, 5, 28, 29]. In [3], the notion of Hilbert indexing is generalized to arbitrary dimensions. The Hilbert SFC is structurally analyzed, which helps in understanding how the Hilbert SFC is built in the multi-dimensional space. [5] studies the properties of several space filling curves in the two- and three-dimensional spaces, and introduces new measures to describe the behavior of any space-filling curve. The notion of irregularity is presented in [28] as a quantitative measure of how irregular a space-filling curve is. In [29], the clustering properties of the Hilbert SFC is analyzed by deriving closed formulas for the number of clusters in a given query region.

Numerous algorithms are developed for efficiently generating different space-filling curves. Recursive algorithms for generating the Hilbert SFC are proposed in [8, 10, 17, 45] and for the Peano SFC in [10, 45]. A table-driven algorithm for the Peano and Hilbert SFCs is proposed in [17]. An algorithm for computing the order of any point in the Hilbert, Peano, and Gray SFCs is proposed in [15]. For a comparison of

P_i	The ith point in a space-filling curve
$P_i.u_k$	The kth dimension in the ith point in a space-filling curve
$Jump(k, N, D)$	The number of *Jump* segments in dimension k in a D-dimensional space with grid size N
$Contiguity(k, N, D)$	The number of *Contiguity* segments in dimension k in a D-dimensional space with grid size N
$Reverse(k, N, D)$	The number of *Reverse* segments in dimension k in a D-dimensional space with grid size N
$Forward(k, N, D)$	The number of *Forward* segments in dimension k in a D-dimensional space with grid size N
$Still(k, N, D)$	The number of *Still* segments in dimension k in a D-dimensional space with grid size N
$J_T(N, D)$	The total number of *Jump* segments in all dimensions in a D-dimensional space with grid size N
$C_T(N, D)$	The total number of *Contiguity* segments in all dimensions in a D-dimensional space with grid size N
$R_T(N, D)$	The total number of *Reverse* segments in all dimensions in a D-dimensional space with grid size N
$F_T(N, D)$	The total number of *Forward* segments in all dimensions in a D-dimensional space with grid size N
$S_T(N, D)$	The total number of *Still* segments in all dimensions in a D-dimensional space with grid size N
V_T	The total description vector $V_T = (J_T, C_T, R_T, F_T, S_T)$

Table 1: Symbols used in the paper.

different space-filling curves, a reader is referred to [1, 5, 13, 37].

3 Segment Types in Space-filling Curves

A D-dimensional space-filling curve with grid size N has N^D-1 segments that connect N^D points. Each segment is classified as one or more of five segment types: *Jump*, *Contiguity*, *Reverse*, *Forward*, and *Still*. In this section, we give a precise definition of each segment type along with an iterative equation to compute the number of segments from each type for each dimension in the multi-dimensional space. For the rest of the paper, we use the notations and definitions given in Table 1.

3.1 Jump

Definition 1 *A* Jump *in an SFC is said to happen when the distance, along any of the dimensions, between two consecutive points in the SFC is greater than one.*

Formally, for any two consecutive multi-dimensional points P_i and P_{i+1} in an SFC, a *Jump* occurs in dimension k iff $abs(P_i.u_k - P_{i+1}.u_k) > 1$. The total number of *Jump* segments in a dimension k in a D-dimensional space with grid size N is: $Jump(k, N, D) = \sum_{i=0}^{N^D-1} f_J(i, k)$ where $f_J(i, k) = 1$ iff $abs(P_i.u_k - P_{i+1}.u_k) > 1$ and 0 otherwise. The total number of *Jump* segments in an SFC is: $J_T(N, D) = \sum_{k=0}^{D-1} Jump(k, N, D)$.

A *Jump* in a space-filling curve reflects the locality of the consecutive points in the order implied by the space-filling curve. For example, consider the Sweep SFC (Figure 1a). By the end of each horizontal sweep, the Sweep SFC jumps back to the beginning of the horizontal axis. Thus, the last point in a horizontal sweep and the first point in the next horizontal sweep will be neighbors in the one-dimensional domain while they are not neighbors in the multi-dimensional space. In contrast, consider the C-Scan and Hilbert SFCs, where they do not have any *Jump* segments. So, any two neighbors in the one-dimensional ordering are guaranteed to be neighbors in the multi-dimensional space. Generally, the lack of *Jump* segments indicates more ability for clustering. However, *Jump* may or may not be a favorable property based on the application type. For example, in a disk-head scheduling [4], *Jumps* are considered bad, as they result in a longer seek time without retrieving any data. On the other side, in multi-priority scheduling, *Jumps* are considered good, as the ability of fast moving among different priority types is required.

3.2 Contiguity

Definition 2 *A* Contiguity *in an SFC is said to happen when the distance, along any of the dimensions, between two consecutive points in the SFC is equal to one.*

Formally, for any two consecutive multi-dimensional points P_i and P_{i+1} in an SFC, a *Contiguity* occurs in dimension k iff $abs(P_i.u_k - P_{i+1}.u_k) = 1$. The total number of *Contiguity* segments in a dimension k in a D-dimensional space with grid size N is: $Contiguity(k, N, D) = \sum_{i=0}^{N^D-1} f_C(i, k)$ where $f_C(i, k) = 1$ iff $abs(P_i.u_k - P_{i+1}.u_k) = 1$ and 0 otherwise. The total number of *Contiguity* segments in an SFC is: $C_T(N, D) = \sum_{k=0}^{D-1} Contiguity(k, N, D)$.

Contiguity reflects the ability of a space-filling curve to go continuously along any of the dimensions. For example, consider the Scan SFC (Figure 1b) where it always go continuously in one of the dimensions. It starts by seven continuous horizontal segments followed by one continuous segment vertically, then another set of continuous horizontal segments. A high ratio of *Contiguity* indicates a lower ratio in *Jump*. As in *Jumps*, *Contiguity* may or may not be favorable, depending on the underlying application.

3.3 Reverse

Definition 3 *A segment in an SFC is termed a* Reverse *segment if the projection of its two consecutive points, along any of the dimensions, results in scanning the dimension in decreasing order.*

Formally, for any two consecutive multi-dimensional points P_i and P_{i+1} in an SFC, a *Reverse* segment occurs in dimension k iff $P_{i+1}.u_k < P_i.u_k$. The total number of *Reverse* segments in a dimension k in a D-dimensional space with grid size N is: $Reverse(k, N, D) = \sum_{i=0}^{N^D-1} f_R(i, k)$ where $f_R(i, k) = 1$ iff $P_{i+1}.u_k < P_i.u_k$ and 0 otherwise. The total number of *Reverse* segments in an SFC is: $R_T(N, D) = \sum_{k=0}^{D-1} Reverse(k, N, D)$.

A *Reverse* segment is also classified as either a *Jump* or a *Contiguity* one. For example, in the Sweep SFC, moving from the first horizontal sweep to the second one is done by a reverse and jump segment. On the other side, moving from the first horizontal sweep to the second one in the Scan SFC is done by seven reverse and continuous segments. Whether reverse segments are favorable or not relates to the semantic of the sorted parameter. For example, consider real-time applications. When applying a space-filling curve to a deadline parameter, the sorting from the largest to the smallest, i.e., in reverse order, means that we visit the points with larger deadline before the points with smaller deadline. In this case, reverse ordering is considered unfavorable. As another example, consider the case of disk-head scheduling [4]. Based on the disk-head movement, alternating between forward and reverse orderings is favorable. In summary, it is important to point out and quantify whether or not a space-filling curve exhibits reverse ordering in its dimensions.

3.4 Forward

Definition 4 *A segment in an SFC is termed a* Forward *segment if the projection of its two consecutive points, along any of the dimensions, results in scanning the dimension in increasing order.*

Formally, for any two consecutive multi-dimensional points P_i and P_{i+1} in an SFC, a *Forward* segment occurs in dimension k iff $P_{i+1}.u_k > P_i.u_k$. The total number of *Forward* segments in a dimension k in a D-dimensional space with grid size N is: $Forward(k, N, D) = \sum_{i=0}^{N^D-1} f_F(i,k)$ where $f_F(i,k) = 1$ iff $P_{i+1}.u_k > P_i.u_k$ and 0 otherwise. The total number of *Forward* segments in an SFC is: $F_T(N, D) = \sum_{k=0}^{D-1} Forward(k, N, D)$.

As in *Reverse* segment, a *Forward* segment is also classified as either a *Jump* or a *Contiguity* segment. For example, the first horizontal sweep in the Sweep SFC have seven forward and continuous segments. On the other side, in the Peano SFC (Figure 1c), the segment that connects the second and the third quadrants is considered as a forward and jump segment in the horizontal dimension. However, it is considered as a reverse and continuous segment in the vertical dimension. A higher ratio of *Reverse* segments indicates a lower ratio of *Forward* segments.

3.5 Still

Definition 5 *A segment in an SFC is termed a* Still *segment when the distance, along any of the dimensions, between the segment's two consecutive points in the SFC is equal to zero.*

Formally, for any two consecutive multi-dimensional points P_i and P_{i+1} in an SFC, a *Still* segment occurs in dimension k iff $P_{i+1}.u_k = P_i.u_k$. The total number of *Still* segments in a dimension k in a D-dimensional space with grid size N is: $Still(k, N, D) = \sum_{i=0}^{N^D-1} f_S(i,k)$ where $f_S(i,k) = 1$ iff $P_{i+1}.u_k = P_i.u_k$ and 0 otherwise. The total number of *Still* segments in an SFC is: $S_T(N, D) = \sum_{k=0}^{D-1} Still(k, N, D)$.

A segment is considered as a *Still* segment if it does not match any of the other types. *Still* segments is the closure of other types. For example, a segment that is neither a *Jump* nor a *Contiguity* is considered as a *Still*. Also, a segment that is neither a *Reverse* nor a *Forward* segment is considered as a *Still* segment. In general, the number of *Still* segments in a dimension k indicates the percent that this dimension

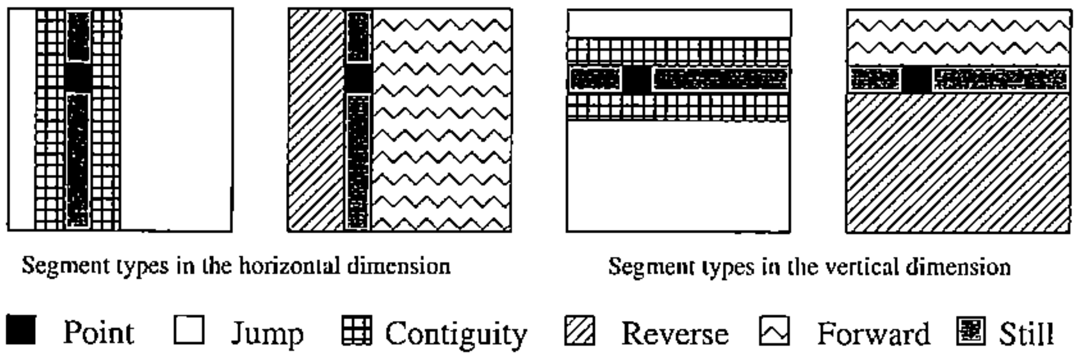

Segment types in the horizontal dimension
Segment types in the vertical dimension
Point
Jump
Contiguity
Reverse
Forward
Still

Figure 3: The relation between segment types.

is neglected to visit other dimensions. For example, consider the Sweep SFC, each horizontal sweep has seven segments that are continuous and forward in the horizontal dimension. However, they are considered as *Still* segments in the vertical dimension. This high ratio of *Still* segments in the vertical dimension in indicates that the Sweep SFC neglects advancing in the vertical dimension in favor of advancing in the horizontal dimension. Unlike other segment types, a *Still* segment cannot be classified as another segment type.

3.6 Relation between segment types

The five segment types can be divided into two categories. The first category, termed the *distance category*, is concerned with the segment length. This includes *Jump*, *Contiguity*, and *Still* segments where the segment length in greater than, equal, or less than one, respectively. The second category, termed the *direction category* is concerned with the direction of the segment. This includes *Reverse*, *Forward*, and *Still* segments. Notice that the *Still* segments belong to the two categories where it serves as the closure of each property. Figure 3 illustrates the difference between the distance category segments and the direction category segments for both the horizontal and vertical dimensions in the two-dimensional space. The relationships among the segment types are summarized in the following Lemma.

Lemma 1 *For any dimension k in a D-dimensional space with grid size N, the following equalities always hold.*

$$Jump(k,N,D) + Contiguity(k,N,D) + Still(k,N,D) = N^D - 1$$
$$Reverse(k,N,D) + Forward(k,N,D) + Still(k,N,D) = N^D - 1$$
$$J_T + C_T + S_T = D(N^D - 1)$$
$$R_T + F_T + S_T = D(N^D - 1)$$

Proof: The proof is given in Appendix A.1. □

From Lemma 1, we deduce the following Corollary:

Corollary 1 *To compute the description vector V, it is enough to compute only three segment types with at least one from each category. The other two segment types can be computed from Lemma 1.*

4 Case Studies

The time complexity for calculating the number of segments of any type in a D-dimensional space with grid size N is $O(N^D)$. Consider the case of 20 dimensions with grid size 16, we need 16^{20} operations to compute the number of *Jumps* of a space-filling curve. To avoid this excessive operation, we derive closed formulas that compute the number of segments of each type for any dimension k in a D-dimensional space with grid size N. In this paper, we concentrate on two non-recursive space-filling curves: the Sweep and Scan SFCs; and three recursive space-filling curves: the Peano, Gray, and Hilbert SFCs. For each space-filling curve, we derive two formulas; the first formula gives the number of segment types in each dimension k, and the second formula gives the total description vector V_T that represents the total number of each segment for all dimensions. Given that the total number of segments in the D-dimensional space is $D(N^D - 1)$, therefore, the percentages of each segment type are computed in the description vector $V = V_T/D(N^D - 1)$.

4.1 Case Study I: The Sweep SFC

Figures 1a and 2a give the Sweep SFC in the two- and three-dimensional spaces with grid sizes eight and four, respectively. The simplicity of the Sweep SFC is the main reason to its wide spread. Applications of the Sweep SFC include storing multi-dimensional arrays in memory and disk scheduling. A D-dimensional Sweep SFC with grid size N is represented by a D digits number in the base N system. The rightmost digit represents the last dimension ($k = D - 1$), while the leftmost digit represents the first dimension ($k = 0$). This means that in order to increase the value of dimension k from v to $v + 1$, the Sweep SFC goes through all the points 0 to N-1 in dimension $k - 1$. We call this event a *Cycle* of the Sweep SFC. For example, in Figure 1a, in order to advance one value in the vertical dimension, the Sweep SFC should go through a *Cycle* from 0 to 7 in the horizontal dimension. The first dimension in the Sweep SFC has N^{D-1} cycles, each with N points. Generally, the kth dimension has N^{D-k-1} cycles, each with N^{k+1} points. Notice that the last dimension has only one cycle that includes all space points (N^D). Table 2 gives an example of computing the Sweep order for the two- and three-dimensional points with a grid size of eight points in each dimension.

Lemma 2 *In a D-dimensional space with grid size N, the number of Jump, Contiguity, Reverse, Forward,*

Point	Octal Number	Convertion Process	Sweep Order	Point	Octal Number	Cpnversion Process	Sweep Order
(2,1)	$(21)_8$	$2 \times 8 + 1$	17	(0,1,3)	$(013)_8$	$0 \times 64 + 1 \times 8 + 3$	11
(5,3)	$(53)_8$	$5 \times 8 + 3$	17	(2,1,4)	$(214)_8$	$2 \times 64 + 1 \times 8 + 4$	140
(7,0)	$(70)_8$	$7 \times 8 + 0$	56	(7,0,7)	$(707)_8$	$7 \times 64 + 0 \times 8 + 7$	455

Table 2: An Example of two- and three-dimensional Sweep SFC with grid size 8 in each dimension.

and Still segments in any dimension k for the Sweep SFC is:

$$\begin{aligned} Jump(k,N,D) &= Reverse(k,N,D) = N^{D-k-1} - 1 \\ Contiguity(k,N,D) &= Forward(k,N,D) = N^{D-k-1}(N-1) \\ Still(k,N,D) &= N^D - N^{D-k} \end{aligned}$$

Proof: The proof is given in Appendix A.2. □

Lemma 3 *The total description vector V_T for the D-dimensional Sweep SFC with grid size N is $V_T = (J_T, C_T, R_T, F_T, S_T)$ where:*

$$\begin{aligned} J_T &= R_T = \frac{N^D - 1}{N - 1} - D \\ C_T &= F_T = N^D - 1 \\ S_T &= DN^D - \frac{N\left(N^D - 1\right)}{N - 1} \end{aligned}$$

The description vector $V = V_T / D(N^D - 1)$.

Proof: The proof is given in Appendix A.3. □

4.2 Case Study II: The Scan SFC

The Scan SFC (Figures 1b and 2b) is a slight modification of the original Sweep SFC. The main motivation is to avoid the *Jump* segments in the Sweep SFC. Thus instead of having one *Jump* and *Reverse* segment between each Sweep *Cycle*, the Scan SFC replaces this segment by a sequence of $N - 1$ *Contiguity* and *Reverse* segments. The Scan SFC have the same concept of a *Cycle* as in the Sweep SFC. However, the Scan SFC distinguishes between even-numbered and odd-numbered cycles. Notice that for the kth dimension, the Scan SFC has N^{D-k-1} cycles. Even-numbered cycles are exactly the same as the Sweep SFC. However, the odd-numbered *Cycles* in the case of the Scan SFC consists of $N - 1$ *Contiguity* and *Reverse* segments rather than *Contiguity* and *Forward* segments as in the case of the Sweep SFC. Also, the transition between each cycle is performed by a *Still* segment in the case of the Scan SFC rather than by a *Jump* segment as in the case of the Sweep SFC. Many applications benefit from the no *Jump* property of the Scan SFC.

Lemma 4 *In a D-dimensional space with grid size N, the number of Jump, Contiguity, Reverse, Forward, and Still segments in any dimension k for the Scan SFC is:*

$$\begin{aligned}
Jump(k, N, D) &= 0 \\
Contiguity(k, N, D) &= N^{D-k-1}(N-1) \\
Still(k, N, D) &= N^{D-k-1}\left(N^{k+1} - N + 1\right) - 1 \\
Reverse(D-1, N, D) &= 0 \\
Reverse(k, N, D) &= \frac{1}{2}N^{D-k-1}(N-1), && k < D-1 \\
Forward(D-1, N, D) &= N - 1 \\
Forward(k, N, D) &= \frac{1}{2}N^{D-k-1}(N-1), && k < D-1
\end{aligned}$$

Proof: The proof is given in Appendix A.4. $\square$

Lemma 5 *The total description vector V_T for the D-dimensional Scan SFC with grid size N is $V_T = (J_T, C_T, R_T, F_T, S_T)$ where:*

$$\begin{aligned}
J_T &= 0 \\
C_T &= N^D - 1 \\
S_T &= (D-1)\left(N^D - 1\right) \\
R_T &= \frac{N}{2}\left(N^{D-1} - 1\right) \\
F_T &= \frac{N}{2}\left(N^{D-1} - 1\right) + N - 1
\end{aligned}$$

The description vector $V = V_T/D(N^D - 1)$.

Proof: The proof is given in Appendix A.5. $\square$

4.3 Case Study III: The Peano SFC

The Peano SFC (Figures 1c and 2c) is introduced by Peano [36] and is also called Morton encoding [31], quad code [16], bit-interleaving [41], N-order [43], locational code [2], or Z-order [34]. The Peano SFC is constructed recursively as in Figure 4. The basic step (Figure 4a) contains four points in the four quadrants of the space. Each quadrant is represented by two binary digits. The most significant digit is represented

Point	Dimensions		Bit	Decimal	Point	Dimensions			Bit	Decimal
	0	1	Interleaving	Order		0	1	2	Interleaving	Order
(2,1)	**010**	001	001001	9	(0,1,3)	**000**	*001*	011	000001011	11
(5,3)	**101**	011	100111	39	(2,1,4)	**010**	*001*	100	001100010	98
(7,0)	**111**	000	101010	42	(7,0,7)	**111**	*000*	111	101101101	365

Table 3: An Example of two- and three-dimensional Peano orders with grid size 8 in each dimension.

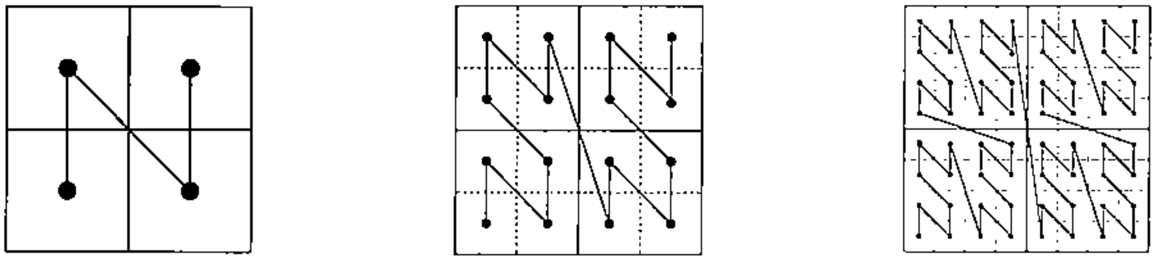


Figure 4: The Peano SFC.

by its x position and the least significant digit is represented by its y position. The Peano SFC orders these points in ascending order (00, 01, 10, 11). Figure 4b contains four repeated blocks of Figure 4a at a finer resolution and is visited in the same order as in Figure 4a. Similarly, Figure 4c contains four repeated blocks of Figure 4b at a finer resolution.

To extend the Peano SFC to the multi-dimensional space, we present the idea of bit-interleaving in the two-dimensional space as shown in Figure 5. Each point in the space is assigned a binary number that results from interleaving bits of the two dimensions. The bits are interleaved according to an interleaving vector T_v=(0,1,0,1). This corresponds to taking the first and third bits from dimension 0 (x) and taking the second and fourth bits from dimension 1 (y). For a D-dimensional space with four points in each dimension, the interleaving vector is $(0, 1, 2, \ldots, D-1, 0, 1, 2, \ldots, D-1)$. For a grid size of N points in each dimension, the term $0, 1, 2, \ldots, D-1$ is repeated $LogN$ times. The points are visited in ascending order according to their binary number representation. Table 3 gives an example of computing the Peano order for two- and three-dimensional points with a grid size of eight points in each dimension.

Lemma 6 *In a D-dimensional space with grid size N, the number of Jump, Contiguity, Reverse, Forward,*

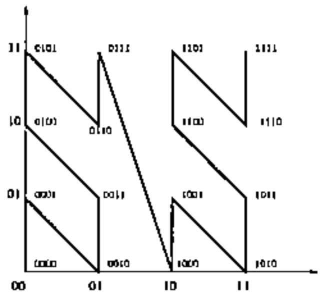


Figure 5: Bit Interleaving in Peano SFC.

and Still segments in any dimension k for the Peano SFC is:

$$\begin{aligned}
Jump(k,N,D) &= \frac{\left(N^D - 2^{2D}\right)\left(2^D - 2\right)}{2^{2D-k}\left(2^D - 1\right)} + 2^k - 1 \\
Contiguity(k,N,D) &= \frac{1}{2^{2D-k}} N^D \left(2^{D+1} - 1\right) + \frac{1}{2^{2D-k}} \frac{N^D - 2^{2D}}{2^D - 1} \\
Still(k,N,D) &= N^D \left(1 - 2^{k-D+1}\right) \\
Reverse(k,N,D) &= \frac{2^k \left(N^D - 2^D\right)\left(2^D - 2\right)}{2^D \left(2^D - 1\right)} + 2^k - 1 \\
Forward(k,N,D) &= \frac{2^k \left(N^D + 2^D - 2\right)}{2^D - 1}
\end{aligned}$$

Proof: The proof is given in Appendix A.6. □

Lemma 7 *The total description vector V_T for the D-dimensional Peano SFC with grid size N is $V_T = (J_T, C_T, R_T, F_T, S_T)$ where:*

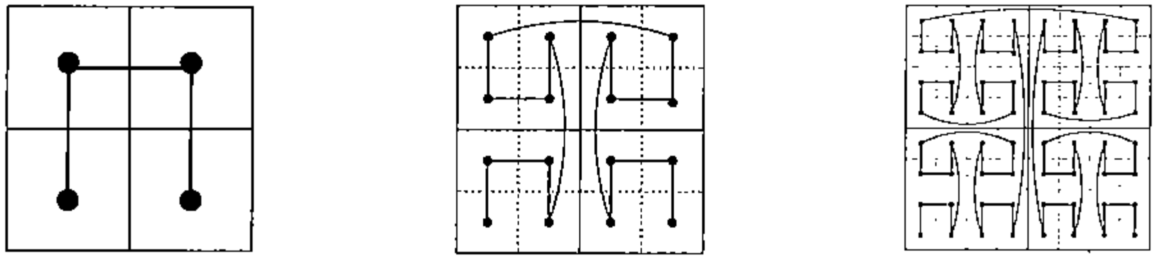
$$\begin{aligned}
J_T &= \left(\frac{N}{2}\right)^D \left(1 - 2^{1-D}\right) + 1 - D \\
C_T &= \left(\frac{N}{2}\right)^D \left(2^{D+1} + 2^{1-D} - 3\right) \\
R_T &= N^D \left(1 - 2^{1-D}\right) + 1 - D \\
F_T &= N^D - 1 \\
S_T &= N^D \left(2^{1-D} + D - 2\right)
\end{aligned}$$

The description vector $V = V_T / D(N^D - 1)$.

Proof: The proof is given in Appendix A.7. □

Point	Dimensions		Bit	Decimal	Point	Dimensions			Bit	Decimal
	0	1	Interleaving	Order		0	1	2	Interleaving	Order
(2,1)	**011**	001	001011	13	(0,1,3)	**000**	*001*	010	000001010	12
(5,3)	**111**	010	101110	52	(2,1,4)	**011**	*001*	110	001101110	75
(7,0)	**100**	000	100000	63	(7,0,7)	**100**	*000*	100	100000100	384

Table 4: An Example of two- and three-dimensional Gray orders with grid size 8 in each dimension.



(a) (b) (c)

Figure 6: The Gray SFC.

4.4 Case Study IV: The Gray SFC

The Gray SFC (Figures 1d and 2d) uses the Gray code representation [18] in contrast to the binary code representation as in the Peano SFC. Figure 6 gives the recursive construction of the Gray SFC. The basic step (Figure 6a) contains four points in the four quadrants of the space. As in the Peano SFC, each quadrant is represented by two binary digits. The most significant digit is represented by its x position and the least significant digit is represented by its y position. The Gray SFC orders these points in ascending order according to the Gray code (00, 01, 11, 10). Figure 6b contains four repeated blocks of Figure 6a at a finer resolution and is visited in Gray order.

Unlike the Peano SFC, the first and the fourth blocks have the same orientation as those of Figure 6a, while the second and the third blocks are constructed by rotating the block of Figure 6a by 180^0. Similarly, Figure 6c is constructed from two blocks of Figure 6b at a finer resolution and two blocks of the rotation of Figure 6b by 180^0. For details about extending the Gray SFC to multi-dimensional space, the reader is referred to [28].

To extend the Gray SFC to the multi-dimensional space, we use the same idea of bit interleaving as in the Peano SFC. Figure 7 gives the bit interleaving in the two-dimensional space with four points in each dimension. Table 4 gives an example of computing the Gray order for two- and three-dimensional points with grid size eight (i.e., eight points) in each dimension.

Lemma 8 *In a D-dimensional space with grid size N, the number of Jump, Contiguity, Reverse, Forward,* *and Still segments in any dimension k for the Gray SFC is:*

$$\begin{aligned}
Jump(k,N,D) &= \frac{(N^D - 2^D)}{2^{D-k}\,(2^D-1)} \\
Contiguity(k,N,D) &= \frac{N^D}{2^{D-k}} \\
Still(k,N,D) &= \frac{(N^D-1)\,(2^D-2^k-1)}{2^D-1} \\
Reverse(0,N,D) &= \frac{N^D-2^D}{2\,(2^D-1)} \\
Reverse(k,N,D) &= \frac{2^{k-1}\,(N^D-1)}{2^D-1}, & k > 0 \\
Forward(0,N,D) &= \frac{N^D-2^D}{2\,(2^D-1)}+1 \\
Forward(k,N,D) &= \frac{2^{k-1}\,(N^D-1)}{2^D-1}, & k > 0
\end{aligned}$$

Proof: The proof is given in Appendix A.8. □

Lemma 9 *The total description vector V_T for the D-dimensional Gray SFC with grid size N is $V_T = (J_T, C_T, R_T, F_T, S_T)$ where:*

$$\begin{aligned}
J_T &= \left(\frac{N}{2}\right)^D - 1 \\
C_T &= \left(\frac{N}{2}\right)^D (2^D-1) \\
R_T &= \frac{N^D-2}{2} \\
F_T &= \frac{N^D}{2} \\
S_T &= (D-1)(N^D-1)
\end{aligned}$$

The description vector $V = V_T/D(N^D-1)$.

Proof: The proof is given in Appendix A.9. □

4.5 Case Study V: The Hilbert SFC

Figure 8 gives the recursive construction of the Hilbert SFC. The basic block of the Hilbert SFC (Figure 8a) is the same as the basic block of the Gray SFC (Figure 6a). The basic block is repeated four times at a finer resolution in the four quadrants, as given in Figure 8b. The quadrants are visited in their gray order. The second and third blocks in Figure 8b have the same orientation as in Figure 8a. The first block is

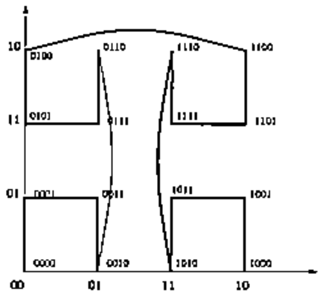


Figure 7: Bit Interleaving in Peano SFC.

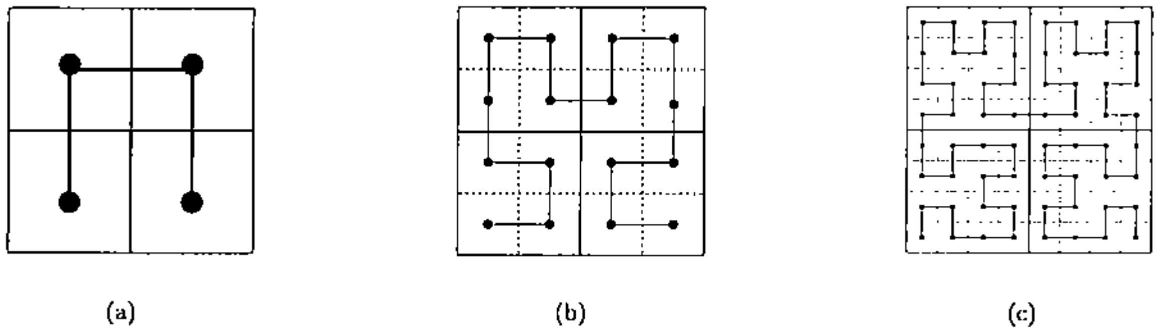


Figure 8: The Hilbert SFC.

constructed from rotating the block of Figure 8a by 90^0, while the fourth block is constructed by rotating the block of Figure 8 by -90^0. Figure 8c is constructed from Figure 8b in an analogous manner.

Lemma 10 *In a D-dimensional space with grid size N, the number of Jump, Contiguity, Reverse, Forward, and Still segments in any dimension k for the Hilbert SFC is:*

$$\begin{aligned}
Jump(k,N,D) &= 0 \\
Contiguity(k,N,D) &= \sum_{i=1}^{D-1} 2^i Contiguity((k+i) \mod D, \frac{N}{2}, D) + 2Contiguity(k, \frac{N}{2}, D) + 2^k \\
Contiguity(k,1,D) &= 0 \\
Still(k,N,D) &= N^D - 1 - Contiguity(k,N,D) \\
Reverse(0,N,D) &= (Contiguity(0,N,D) - N + 1)/2 \\
Reverse(k,N,D) &= Contiguity(k,N,D)/2 \quad k > 0 \\
Forward(k,N,D) &= N^D - 1 - Reverse(k,N,D) - Still(k,N,D)
\end{aligned}$$

Proof: The proof is given in Appendix A.10. □

Lemma 11 *The total description vector V_T for the D-dimensional Hilbert SFC with grid size N is $V_T =$*

$(J_T, C_T, R_T, F_T, S_T)$ *where:*

$$J_T = 0$$
$$C_T = N^D - 1$$
$$R_T = \frac{N}{2}\left(N^{D-1} - 1\right)$$
$$F_T = \frac{N}{2}\left(N^{D-1} + 1\right) - 1$$
$$S_T = (D-1)(N^D - 1)$$

The description vector $V = V_T/D(N^D - 1)$.

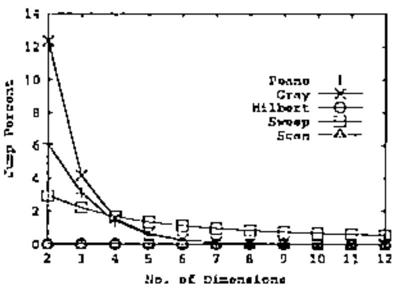
Proof: The proof is given in Appendix A.11. □

5 Performance Evaluation

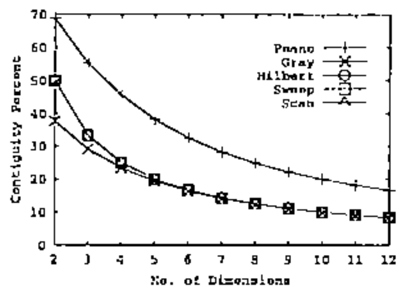
In this section, we perform comprehensive experiments to compare the Sweep, Scan, Peano, Gray, and Hilbert SFCs with respect to the different segment types. The results in this section are computed using the closed formulas developed in Section 4. Notice that it is timely infeasible to compute segment types in high-dimensional spaces using the definition and iterative equations from Section 3.

5.1 Scalability of Space-filling Curves

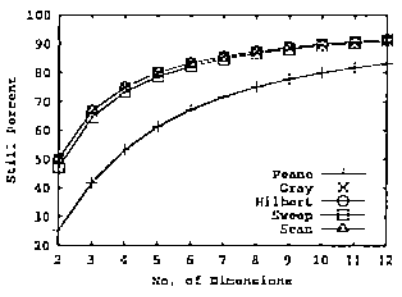
In this section, we address the issue of scalability, i.e., when the number of dimensions and/or the number of points per dimension increase. For the following experiments, we use Lemmas 3, 5, 7, 9, and 11 to compute the description vector V. Figure 9 gives the results of setting the grid size N=16, while measuring different segment types (*Jump*, *Reverse*, and *Still*) up to 12 dimensions. An interesting result appears in the *Jump* segments (Figure 9a) where both the Peano and Gray SFCs have very low percentage (almost 0%) of *Jumps* after six dimensions while the Hilbert and Scan SFCs have no *Jumps* for any dimensions. The fact that the Hilbert SFC has no *Jumps* is well-known [15, 29], and it is the main criteria for why the Hilbert SFC is chosen for many applications e.g., [3, 15, 23]. However, this experiment emphasizes that both the Peano and Gray SFCs share the property of no *Jumps* with the Hilbert SFC for medium and high dimensionality. For *Contiguity*, all space-filling curves almost have the same number of *Contiguity* segments, except the Peano SFC, where it has higher *Contiguity* segments than the other space-filling curves. This affects the number of *Still* segments, where the Peano SFC has the least number of *Still* segments. As it appears from its definition, the Sweep SFC has very low *Reverse* segments, while the Peano SFC has the highest number of *Reverse* segments. For the *Forward* segments, both the Sweep and Peano SFCs have the highest ratio.



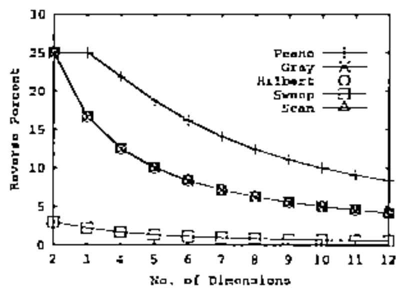
(a) Jump



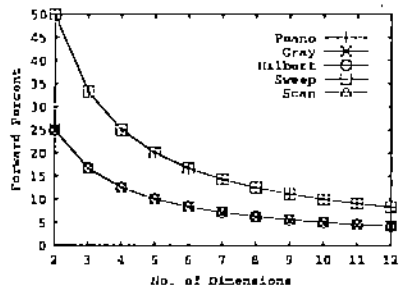
(b) Contiguity



(c) Still



(d) Reverse

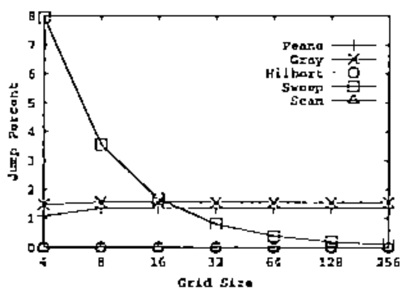


(e) Forward

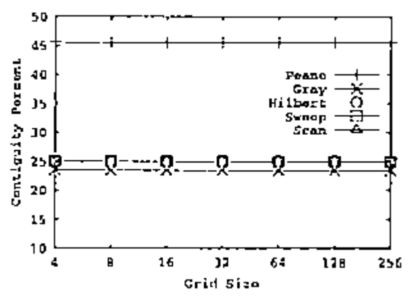
Figure 9: Scalability of space-filling curve w.r.t dimensionality.

The Gray and Hilbert SFCs have similar behavior for all segment types except for low-dimensionality in the *Jump* and *Contiguity* segments. Notice that all segment types except *Still* are decreasing as the number of dimensions increases. The reason for this comes from the *Still* segment definition. A *Still* segment indicates that the value in one of its dimensions does not change. With a larger number of dimensions, it is difficult to find a segment that connects two consecutive multi-dimensional points that are different in all dimensions. Thus, almost each segment is counted as *Still* for one or more dimensions.

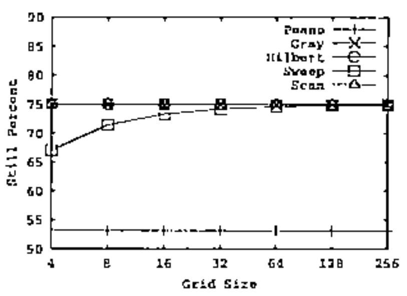
The second set of experiments (Figure 10) tests the four-dimensional space with grid size up to 256. All space-filling curves except the Sweep SFC almost have constant percentage regardless of the grid size. This can be noted from the description vector V, where getting the $\lim_{N\to\infty} V$ gives a constant value that does not depend on N. An interesting result is that the Scan and Hilbert SFCs have the same performance for all segment types. The Gray SFC share the same performance with the Hilbert and Scan SFCs for the *Reverse*, *Forward*, and *Still* segments. However, the Gray SFC has more *Jumps* and lower *Contiguity* than the Hilbert SFC. The Peano SFC has the highest ratio, with a large margin, of both *Contiguity* and *Reverse* segments. This is balanced by the very low ratio of *Still* segments in the Peano SFC. The Sweep SFC is the only space-filling curve that is affected by the change of grid size. However, it tends to be stable after grid size 64.



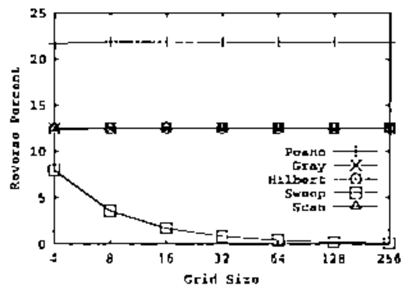
(a) Jump



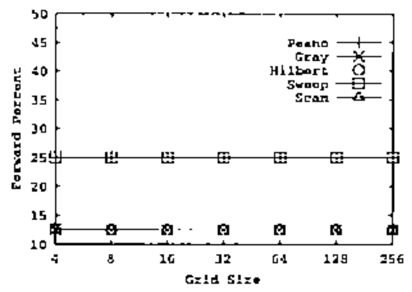
(b) Contiguity



(c) Still



(d) Reverse



(e) Forward

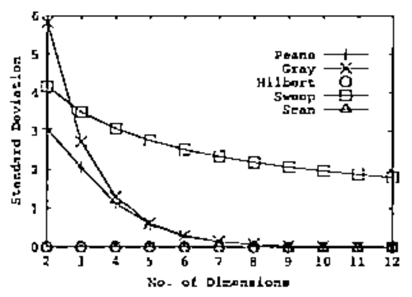
Figure 10: Scalability of space-filling curves w.r.t grid size.

5.2 Fairness of Space-filling Curves

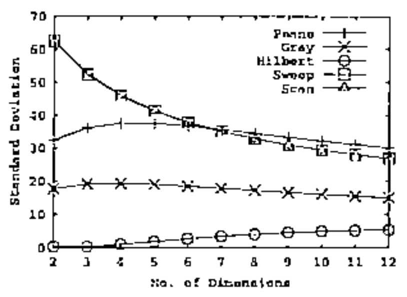
In this section, we test the fairness[1] of space-filling curves. For each segment type T, we use the standard deviation of the number of T segments over all dimensions as an indication for fairness. The lower the standard deviation the more fair the space-filling curve is. For the experiments of this section, we use Lemmas 2, 4, 6, 8, and 10 to compute the number of segments for each segment type over each individual dimension rather than the total that is used in the description vector.

Figure 11 gives the standard deviation for all segment types for up to the 12-dimensional space with grid size 16. It is clear that for all segment types, the Hilbert SFC is the most fair space-filling curve with very low standard deviation. In general, recursive space-filling curves tend to be more fair than non-recursive space-filling curves. This comes from the fact that the recursive space-filling curves divide the space into equal fragments. Each fragment is dealt with in the same way. An exception is the *Reverse* segments in the Sweep SFC, where it has very low standard deviation. This comes from the very low number of *Reverse* segments in all dimensions of the Sweep SFC. Among the recursive space-filling curve, the Peano SFC gives the worst performance. The interesting result is that both the Peano and Gray SFCs tend to be more fair as the dimensionality increases while the Hilbert SFC behaves the opposite. This

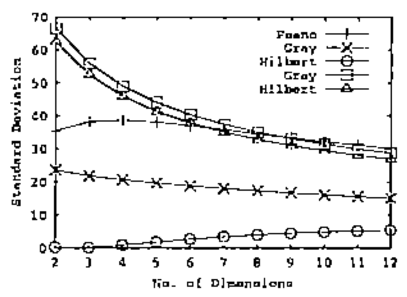
[1]We say that a space-filling curve is fair if it has similar behavior towards all dimensions in the multi-dimensional space.



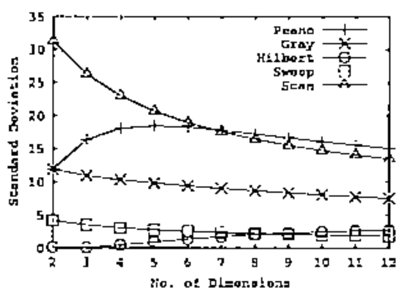
(a) Jump



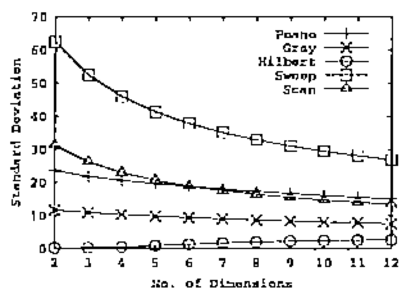
(b) Contiguity



(c) Still



(d) Reverse



(e) Forward

Figure 11: Fairness of space-filling curves.

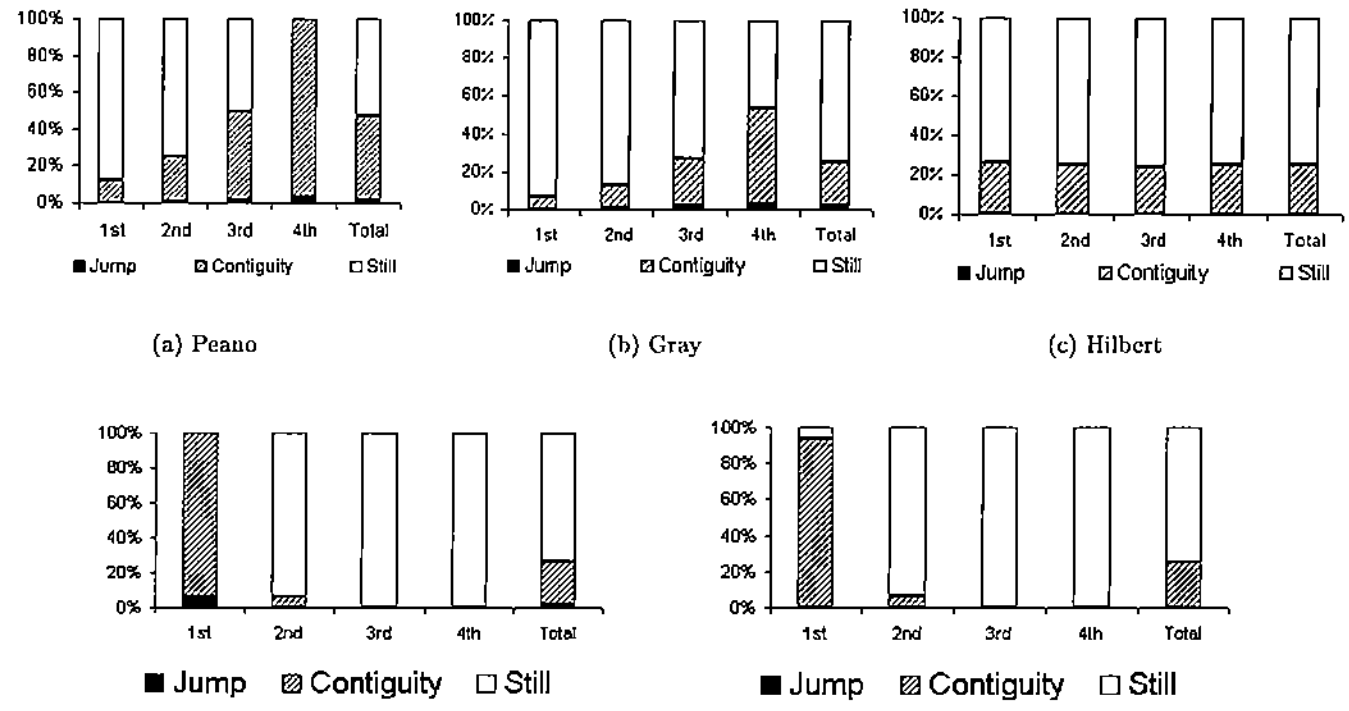
indicates that for very high dimensionality, the Hilbert SFC may not be the most fair space-filling curve.

5.3 Intentional Bias of Space-filling Curves

A very critical point for SFC-based applications is how to assign the different parameters to the space dimensions. In this section, we explore the intentional bias[2] of each space-filling curve by plotting its behavior for each dimension individually. Figures 12 and 13 give the intentional bias for distance (*Jump, Contiguity, Still*) and direction segments (*Reverse, Forward, Still*), respectively. The experiment is performed for the four-dimensional space with grid size 16. Each dimension is plotted individually as a stacked bar that contains the percentage of distance or direction segments. The fifth column is the percentage of the total number of segments over all dimensions from each type. Note that the height of each bar is 100 (refer to Lemma 1).

From Figure 12, the percentage of *Jumps* in the Peano, Gray, and Sweep SFCs is negligible. The Hilbert SFC is not biased to any dimension. This agrees with the result in the previous section, where the Hilbert SFC has a very low standard deviation. With respect to *Contiguity*, the Peano SFC is biased

[2]We say that an SFC is intentionally biased towards a certain dimension k with respect to segment type T if the SFC has more T segments in dimension k with respect to all other dimensions



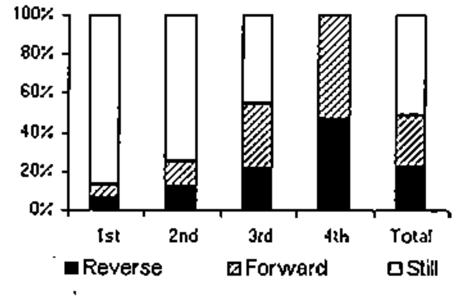
(a) Peano (b) Gray (c) Hilbert

(d) Sweep (e) Scan

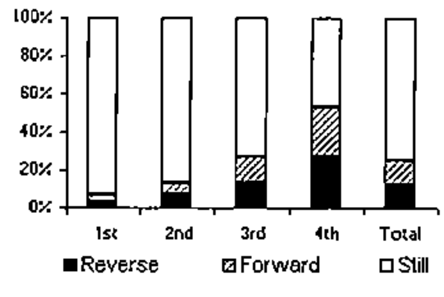
Figure 12: Intentional bias of space-filling curves w.r.t distance segments.

towards the last dimension where almost all the segments are *Contiguity* segments with no *Still* segments. With the increase of the dimension number k, the number of the *Contiguity* segments is increasing rapidly, and the number of *Still* segments is decreasing. The Gray SFC has similar behavior as in the Peano SFC, however, the increase/decrease in *Contiguity/Still* segments is slower. On the other hand, the Sweep and Scan SFCs have very high *Contiguity* segments in the first dimension followed by a very low *Contiguity* segments in the second dimension. There is almost no *Contiguity* in the other dimensions.

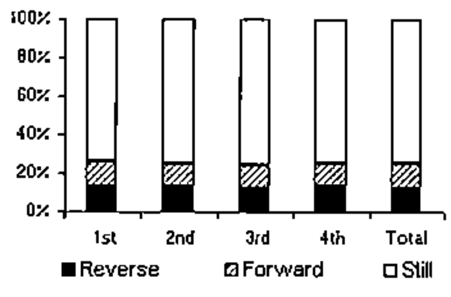
Figure 13 gives the results of the same experiment for direction segments. The same analysis is applied, where the Hilbert SFC is extremely fair, while the Peano SFC is biased towards the last dimension. The only difference here, is that the bias of the Peano and Gray SFCs is with respect to both the *Reverse* and *Forward* segments instead of only the *Jump* segments in Figure 12. Note that in the three recursive SFCs, the percentages of the *Reverse* and *Forward* segments are almost equal for all dimensions. On the other hand, the non-recursive SFCs almost have only *Still* segments after the second dimension. This is the main reason why non-recursive SFCs have very high standard deviation in Figure 11. The Sweep SFC has very low number of *Reverse* segments in the first dimension. On the other hand, the number of *Forward* and *Reverse* segments are equal in the Scan SFC.



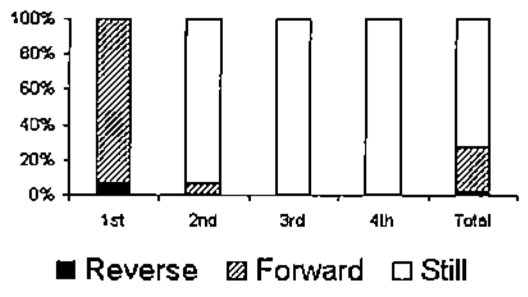
(a) Peano



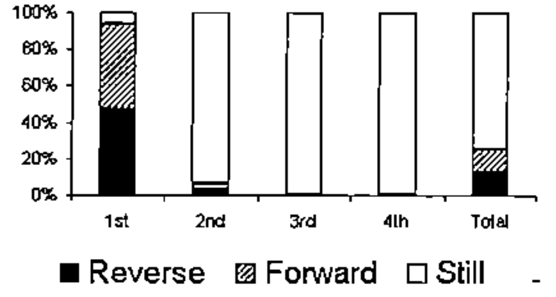
(b) Gray



(c) Hilbert



(d) Sweep



(e) Scan

Figure 13: Intentional bias of space-filling curves w.r.t direction segments.

6 Conclusions

Space-filling curves are used as a mapping scheme from the multi-dimensional space into the one-dimensional space. The behavior of different space-filling curves in the D-dimensional space is analyzed. A description vector V is proposed to give a brief description for each space-filling curve. Closed formulas that depend on the space dimensionality and grid size are derived to compute V. The idea is to divide the space-filling curve into a set of connected segments. Each segment connects two consecutive multi-dimensional points. Five segment types are distinguished, namely, *Jump*, *Contiguity*, *Reverse*, *Forward*, and *Still*. The description vector V contains the percentage of occurrence of each segment type. Several experiments are conducted to show the scalability and fairness of space-filling curves with respect to segment types.

References

[1] David J. Abel and David M. Mark. A comparative analysis of some two-dimensional orderings. *Intl. Journal of Geographical Information Systems*, 4(1):21–31, 1990.

[2] David J. Abel and J. Smith. A data structure and algorithm based on a linear key for a rectangle

retrieval problem. *Computer Vision Graphics Image Processing*, 24:1–13, 1983.

[3] Jochen Alber and Rolf Niedermeier. On multi-dimensional hilbert indexing. In *Proc. of the 4th Intl. Computing and Combinatorics Conference, COCOON*, pages 329–338, Taipei, Taiwan, August 1998.

[4] Walid G. Aref, Khaled El-Bassyouni, Ibrahim Kamel, and Mohamed F. Mokbel. Scalable qos-aware disk-scheduling. In *International Database Engineering and Applications Symposium, IDEAS*, Alberta, Canada, July 2002.

[5] Walid G. Aref and Ibrahim Kamel. On multi-dimensional sorting orders. In *Proc. of the 11th Intl. Conf. on Database and Expert Systems Applications, DEXA*, pages 774–783, London, September 2000.

[6] Tetsuo Asano, Desh Ranjan, Tomas Roos, Emo Welzl, and Peter Widmayer. Space-filling curves and their use in the design of geometric data structures. *Theoretical Computer Science, TCS*, 181(1):3–15, 1997.

[7] Christian Bohm, Gerald Klump, and Hans-Peter Kriegel. *xz*-ordering: A space-filling curve for objects with spatial extensio. In *Proc. of 6th Intl. Symp. on Large Spatial Databases, SSD*, pages 75–90, Hong Kong, July 1999.

[8] Greg Breinholt and Christoph Schierz. Algorithm 781: Generating hilbert's space-filling curve by recursion. *ACM Trans. on Mathematical Software, TOMS*, 24(2):184–189, June 1998.

[9] Thomas Brinkhoff, Hans-Peter Kriegel, and Bernhard Seeger. Efficient processing of spatial joints using r-trees. In *Proc. of the intl. conf. on Management of data, SIGMOD*, pages 237–246, Washington D.C., May 1993.

[10] A. J. Cole. A note on space filling curves. *Software–Practice and Experience, SPE*, 13(12):1181–1189, 1983.

[11] Douglas Comer. The ubiquitous b-tree. *ACM Computing Surveys*, 11(2):121–137, June 1979.

[12] Christos Faloutsos. Gray codes for partial match and range queries. *IEEE Trans. on Software Engineering, TSE*, 14(10):1381–1393, October 1988.

[13] Christos Faloutsos. Analytical results on the quadtree decomposition of arbitrary rectangles. *Pattern Recognition Letters*, 13(1):31–40, January 1992.

[14] Christos Faloutsos and Yi Rong. Dot: A spatial access method using fractals. In *Proc. of Intl. Conf. on Data Engineering, ICDE*, pages 152–159, Kobe, Japan, April 1991.

[15] Christos Faloutsos and Shari Roseman. Fractals for secondary key retrieval. In *Proc. of the 8th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, PODS*, pages 247–252, Philadelphia, March 1989.

[16] Raphael A. Finkel and Jon L. Bentley. Quad trees: a data structure for retrieval on composite keys. *Acta Informatica*, 4:1–9, 1974.

[17] Leslie M. Goldschlager. Short algorithms for space-filling curves. *Software–Practice and Experience, SPE*, 11(1):99–100, 1981.

[18] F. Gray. Pulse code communications. *US Patent 2632058*, 1953.

[19] Antonin Guttman. R-trees: A dynamic index structure for spatial indexing. In *Proc. of the intl. conf. on Management of data, SIGMOD*, pages 47–57, Boston, MA, June 1984.

[20] D. Hilbert. Ueber stetige abbildung einer linie auf ein flashenstuck. *Mathematishe Annalen*, pages 459–460, 1891.

[21] H. V. Jagadish. Linear clustering of objects with multiple attributes. In *Proc. of the intl. conf. on Management of data, SIGMOD*, pages 332–342, Atlantic City, NJ, June 1990.

[22] Ibrahim Kamel and Christos Faloutsos. On packing r-trees. In *Proc. of the 2nd Intl. Conf. on Information and knowledge Management, CIKM*, pages 490–499, Washington D. C., November 1993.

[23] Ibrahim Kamel and Christos Faloutsos. Hilbert r-tree: An improved r-tree using fractals. In *Proc. of the 20th Intl. Conf. on Very Large Data Bases, VLDB*, pages 500–509, Santiago, Chile, September 1994.

[24] Jonathan K. Lawder and Peter. J. H. King. Using space-filling curves for multi-dimensional indexing. In *Proc. of the 17th British National Conf. on Databases, BNCOD*, pages 20–35, UK, July 2000.

[25] Jonathan K. Lawder and Peter. J. H. King. Querying multi-dimensional data indexed using the hilbert space filling curve. *SIGMOD Record*, 30(1), March 2001.

[26] S. Liao, Mario A. Lopez, and Scott.T. Leutenegger. High dimensional similarity search with space-filling curves. In *Proc. of Intl. Conf. on Data Engineering, ICDE*, pages 615–622, Heidelberg, Germany, April 2001.

[27] John M. Mellor-Crummey, David B. Whalley, and Ken Kennedy. Improving memory hierarchy performance for irregular applications. In *Proc. of the Intl. Conf. on Supercomputing, ICS*, pages 425–433, Rhodes, Greece, June 1999.

[28] Mohamed F. Mokbel and Walid G. Aref. Irregularity in multi-dimensional space-filling curves with applications in multimedia databases. In *Proc. of the 2nd Intl. Conf. on Information and knowledge Management, CIKM*, pages 512–519, Atlanta, GA, November 2001.

[29] B. Moon, H. V. Jagadish, Christos Faloutsos, and J. Salz. Analysis of the clustering properties of hilbert space-filling curve. *IEEE Trans. on Knowledge and Data Engineering, TKDE*, 13(1):124–141, 2001.

[30] E. H. Moore. On certain crinkly curves. *Trans. Am. Math Soc.*, pages 72–90, 1900.

[31] G. M. Morton. A computer oriented geodetic data base and a new technique in file sequences. *IBM*, 1966.

[32] Rolf Niedermeier, Klaus Reinhardt, and Peter Sanders. Towards optimal locality in mesh-indexing. In *Proc. of the 11th Intl. Symp. on Fundamentals of Computation Theory, FCT*, pages 364–375, Krakow, Poland, September 1997.

[33] Jack A. Orenstein. Spatial query processing in an object-oriented database system. In *Proc. of the intl. conf. on Management of data, SIGMOD*, pages 326–336, Washington D.C., May 1986.

[34] Jack A. Orenstein and T. H. Merrett. A class of data structures for associative searching. In *Proc. of the 3rd ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, PODS*, pages 181–190, Ontario, Canada, April 1984.

[35] Chao-Wei Ou, Manoj Gunwani, and Sanjay Ranka. Architecture-independent locality-improving transformations of computational graphs embedded in k-dimensions. In *Proc. of the 9th ACM Intl. Conf. on Supercomputing, ICS*, pages 289–298, Barcelona, Spain, July 1995.

[36] G. Peano. Sur une courbe qui remplit toute une air plaine. *Mathematishe Annalen*, 36:157–160, 1890.

[37] H. Sagan. *Space Filling Curves.* Springer, Berlin, 1994.

[38] Kenneth C. Sevcik and Nick Koudas. Filter trees for managing spatial data over a range of size granularities. In *Proc. of the 22th Intl. Conf. on Very Large Data Bases, VLDB*, pages 16–27, Bombay, India, September 1996.

[39] W. Sierpinski. Sur une nouvelle courbe qui remplit toute une aire plaine. *Bull. Acad. Sci. Cracovie, SerieA*, pages 462–478, 1912.

[40] M. Thottethodi, S. Chatterjee, and A.R. Lebeck. Tuning strassan matrix multiplication algorithm for memory efficiency. In *Proc. of SC98: High Performance Computing ad Networking*, Orlando, FL, November 1998.

[41] H. Tropf and H. Herzog. Multidimensional range search in dynamically balanced trees. *Angewandte Informatik*, pages 71–77, 1981.

[42] Luiz Velho and Jonas Gomes. Digital halftoning with space filling curves. *Computer Graphics*, 25(4):81–90, July 1991.

[43] M. White. N-trees: Large ordered indexes for multi-dimensional space. statistical research division. *US Bureau of the Census*, 1980.

[44] I. H. Witten and M. Neal. Using peano curves for bilevel display of continuous tone images. *IEEE Computer Graphics and Applications*, pages 47–52, 1982.

[45] I. H. Witten and B. Wyvill. On the generation and use of space-filling curves. *Software–Practice and Experience*, 3:519–525, 1983.

[46] Y. Zhang and R. E. Webber. Space diffusion: An improved parallel halftoning technique using space-filling curves. *In Computer Graphics Proc.*, pages 305–312, August 1993.

A Appendix

A.1 Proof of Lemma 1

Proof: A D-dimensional space-filling curve with grid size N has N^D points connected by N^D-1 segments. According to the definition of segments in Section 3 and Figure 3, any segment has a distance and a direction. Based on the distance, any segment is classified as either a *Jump*, *Contiguity* or *Still* segment. Therefore,

$$Jump(k,N,D) + Contiguity(k,N,D) + Still(k,N,D) = N^D - 1$$

Based on the direction, any segment is classified as either a *Reverse*, *Forward* or *Still* segment. Therefore,

$$Reverse(k,N,D) + Forward(k,N,D) + Still(k,N,D) = N^D - 1$$

By summing over all dimensions,

$$\sum_{k=0}^{D-1} Jump(k,N,D) + Contiguity(k,N,D) + Still(k,N,D) = \sum_{k=0}^{D-1} (N^D - 1), and$$

$$\sum_{k=0}^{D-1} Reverse(k,N,D) + Forward(k,N,D) + Still(k,N,D) = \sum_{k=0}^{D-1} (N^D - 1)$$

Therefore,

$$J_T + C_T + S_T = D(N^D - 1)$$

$$R_T + F_T + S_T = D(N^D - 1)$$

□

A.2 Proof of Lemma 2

Proof: We start by the first dimension:

$$Jump(0,N,D) = N^{D-1} - 1$$

$$Contiguity(0,N,D) = N^{D-1}(N-1)$$

From the definition of the Sweep SFC, we have the recurrence relations:

$$Jump(k,N,D) = Jump(k-1,N,D-1)$$

$$Contiguity(k,N,D) = Contiguity(k-1,N,D-1)$$

Solving these recurrence relations, therefore:

$$Jump(k, N, D) = N^{D-k-1} - 1$$
$$Contiguity(k, N, D) = N^{D-k-1}(N-1)$$

From Lemma 1, we have: $Still(k, N, D) = N^D - N^{D-k}$. From the definition of the Sweep SFC, every *Jump* segment is counted as a *Reverse* segment, and every *Contiguity* segment is counted as a *Forward* segment. Therefore,

$$Reverse(k, N, D) = N^{D-k-1} - 1, \ and$$
$$Forward(k, N, D) = N^{D-k-1}(N-1).$$

□

A.3 Proof of Lemma 3

Proof: For any segment type X in Lemma 2, X_T is computed from the equation: $X_T = \sum_{k=0}^{D-1} X$. □

A.4 Proof of Lemma 4

Proof: The Scan SFC has no *Jump* segments in all its dimensions, i.e., $Jump(k, N, D) = 0$. The main distinction between the Sweep and Scan SFCs is the direction of the odd-numbered *Cycles*. However, the length of the segments inside each *Cycle* is the same. Thus, the number of *Contiguity* segments is the same in both the Sweep and Scan SFCs. Therefore, $Contiguity(k, N, D) = N^{D-k-1}(N-1)$. From Lemma 1, we have $Still(k, N, D) = N^{D-k-1}\left(N^{k+1} - N + 1\right) - 1$. The *Reverse* segments in the Scan SFC appears only in the odd-numbered *Cycles*. For all dimensions, the number of odd *Cycles* is the same as the number of the even *Cycles*. Thus, the number of *Reverse* segments is the same as the number of the *Forward* segments. Using Lemma 1, we have $2Reverse(k, N, D) = Contiguity(k, N, D)$. Thus, $Reverse(k, N, D) = Forward(k, N, D) = \frac{1}{2}N^{D-k-1}(N-1)$. An exception is the last dimension $k = D-1$. The last dimension has only one *Cycle*. Thus, there are no *Reverse* segments in the last dimension, i.e., $Reverse(D-1, N, D) = 0$. This means that the number of *Forward* segments in the last dimension equals the number of *Contiguity* segments. Therefore, $Forward(D-1, N, D) = N - 1$.

□

A.5 Proof of Lemma 5

Proof: For any segment type X in Lemma 4, X_T is computed from the equation: $X_T = \sum_{k=0}^{D-1} X$. □

A.6 Proof of Lemma 6

Proof: We start by the following base equations:

$$\begin{aligned} Jump(0,4,D) &= 0 \\ Contiguity(0,4,D) &= 2^{D+1}-1 \\ Reverse(0,4,D) &= 2^D-2 \end{aligned}$$

Then, we can construct the following recursive equations for the first dimension ($k = 0$):

$$\begin{aligned} Jump(0,N,D) &= 2^D Jump(0,\frac{N}{2},D) + 2^D - 2 \\ Contiguity(0,N,D) &= 2^D Contiguity(0,\frac{N}{2},D) + 1 \\ Reverse(0,N,D) &= 2^D Reverse(0,\frac{N}{2},D) + 2^D - 2 \end{aligned}$$

By solving these recurrence relations for the first dimension,

$$\begin{aligned} Jump(0,N,D) &= \frac{\left(N^D - 2^{2D}\right)\left(2^D-2\right)}{2^{2D}\left(2^D-1\right)} \\ Contiguity(0,N,D) &= \frac{N^D}{2^{2D}}\left(2^{D+1}-1\right) + \frac{N^D-2^{2D}}{2^{2D}\left(2^D-1\right)} \\ Reverse(0,N,D) &= \frac{\left(2^D-2\right)\left(N^D-2^D\right)}{2^D\left(2^D-1\right)} \end{aligned}$$

For the other dimensions, we have the following recurrence relations:

$$\begin{aligned} Jump(k,N,D) &= 2Jump(k-1,\frac{N}{2},D) + 1 \\ Contiguity(k,N,D) &= 2Contiguity(k-1,\frac{N}{2},D) \\ Reverse(k,N,D) &= 2Reverse(k-1,\frac{N}{2},D) + 1 \end{aligned}$$

By solving the recurrences,

$$\begin{aligned} Jump(k,N,D) &= \frac{\left(N^D - 2^{2D}\right)\left(2^D-2\right)}{2^{2D-k}\left(2^D-1\right)} + 2^k - 1 \\ Contiguity(k,N,D) &= \frac{1}{2^{2D-k}}N^D\left(2^{D+1}-1\right) + \frac{1}{2^{2D-k}}\frac{N^D-2^{2D}}{2^D-1} \\ Reverse(k,N,D) &= \frac{2^k\left(N^D-2^D\right)\left(2^D-2\right)}{2^D\left(2^D-1\right)} + 2^k - 1 \end{aligned}$$

Using Lemma 1, therefore,

$$\mathit{Still}(k,N,D) = N^D\left(1-2^{k-D+1}\right)$$
$$\mathit{Forward}(k,N,D) = \frac{2^k\left(N^D+2^D-2\right)}{2^D-1}$$

□

A.7 Proof of Lemma 7

Proof: For any segment type X in Lemma 6, X_T is computed from the equation: $X_T = \sum_{k=0}^{D-1} X$. □

A.8 Proof of Lemma 8

Proof: We start by the following base equations:

$$\mathit{Jump}(0,4,D) = 1$$
$$\mathit{Contiguity}(0,4,D) = 2^D$$

Then, we can construct the following recursive equations for the first dimension ($k=0$):

$$\mathit{Jump}(0,N,D) = 2^D\mathit{Jump}(0,\frac{N}{2},D)+1$$
$$\mathit{Contiguity}(0,N,D) = 2^D\mathit{Contiguity}(0,\frac{N}{2},D)$$

By solving these recurrence relations for the first dimension,

$$\mathit{Jump}(0,N,D) = \frac{N^D-2^D}{2^D\left(2^D-1\right)}$$
$$\mathit{Contiguity}(0,N,D) = \left(\frac{N}{2}\right)^D$$

For the other dimensions, we have the following recurrence relations:

$$\mathit{Jump}(k,N,D) = 2\mathit{Jump}(k-1,N,D)$$
$$\mathit{Contiguity}(k,N,D) = 2\mathit{Contiguity}(k-1,N,D)$$

By solving the recurrences,

$$\mathit{Jump}(k,N,D) = \frac{\left(N^D-2^D\right)}{2^{D-k}\left(2^D-1\right)}$$
$$\mathit{Contiguity}(k,N,D) = \frac{N^D}{2^{D-k}}$$

Using Lemma 1, therefore

$$Still(k, N, D) = \frac{\left(N^D - 1\right)\left(2^D - 2^k - 1\right)}{2^D - 1}$$

One of the properties of the Gray SFC is that it has the same number of *Reverse* and *Forward* segments for all dimensions, except for the first dimension, where the number of the *Forward* segments is larger by 1. Therefore,

$$\begin{aligned} Forward(0, N, D) &= Reverse(0, N, D) + 1 \\ Forward(k, N, D) &= Reverse(k, N, D), & k > 0 \end{aligned}$$

From Lemma 1, we have:

$$\begin{aligned} Reverse(0, N, D) &= \frac{Jump(0, N, D) + Contiguity(0, N, D) - 1}{2} \\ Reverse(k, N, D) &= \frac{Jump(k, N, D) + Contiguity(k, N, D)}{2}, & k > 0 \end{aligned}$$

Solving these equations results in:

$$\begin{aligned} Reverse(0, N, D) &= \frac{N^D - 2^D}{2\left(2^D - 1\right)} \\ Reverse(k, N, D) &= \frac{2^{k-1}\left(N^D - 1\right)}{2^D - 1} & k > 0 \\ Forward(0, N, D) &= \frac{N^D - 2^D}{2\left(2^D - 1\right)} + 1 \\ Forward(k, N, D) &= \frac{2^{k-1}\left(N^D - 1\right)}{2^D - 1} & k > 0 \end{aligned}$$

□

A.9 Proof of Lemma 9

Proof: For any segment type X in Lemma 8, X_T is computed from the equation: $X_T = \sum_{k=0}^{D-1} X$. □

A.10 Proof of Lemma 10

Proof: As in the Scan SFC, there is no *Jump* segments in the Hilbert SFC, i.e., $Jump(k, N, D) = 0$. The Hilbert SFC of grid size N consists of 2^D blocks of the Hilbert SFC of grid size $N/2$ rotated along the different dimensions. Only two of these blocks are not rotated. Generally, for any dimension $(k + i)$ mod D, there are 2^i blocks rotated along the ith dimension. The 2^D segments that connect different blocks contain 2^k *Contiguity* segments. Therefore, we have the recurrence relation:

$$Contiguity(k,N,D) = \sum_{i=1}^{D-1} 2^i Contiguity((k+i) \mod D, \frac{N}{2}, D) + 2Contiguity(k, \frac{N}{2}, D) + 2^k$$
$$Contiguity(k,1,D) = 0$$

From Lemma 1, $Still(k,N,D) = N^D - 1 - Contiguity(k,N,D)$. As in the Scan SFC, the total number of *Reverse* and *Forward* segments equals the number of *Contiguity* segments. For all dimensions $k > 0$, the number of *Reverse* segments equals the number of *Forward* segments. The reason is that half the rotations of the basic figure of the Hilbert SFC are clockwise and the other half are anticlockwise. Thus, the ratio of the *Reverse* and *Forward* segments is preserved. For example, in Figure 8a, the second dimension (the vertical one) has one *Reverse* and one *Forward* segment. Figure 8b consists of four blocks of Figure 8a. Two of these blocks (the two upper blocks) are not rotated, which results in two *Forward* and two *Reverse* segments. The third block (the lower left block) is rotated clockwise, which results in one *Forward* segment. The fourth (the lower right block) is rotated anticlockwise results in one *Reverse* segment. Thus, the ratio of the *Forward* and *Reverse* segments is preserved with the increase of the grid size. An exception of this is the first dimension $k = 0$, where the number of *Forward* segments is more than the number of *Reverse* segments by $N - 1$. Therefore,

$$Reverse(0,N,D) = (Contiguity(0,N,D) - N + 1)/2$$
$$Reverse(k,N,D) = Contiguity(k,N,D)/2 \quad k > 0$$
$$Forward(k,N,D) = N^D - 1 - Reverse(k,N,D) - Still(k,N,D)$$

□

A.11 Proof of Lemma 11

Proof: For any segment type X in Lemma 10, X_T is computed from the equation: $X_T = \sum_{k=0}^{D-1} X$. □