Generation of Spatial Orders and Space-Filling Curves

Günther Schrack and Leo Stocco, *Member, IEEE*

Keywords: spatial order, space-filling curve, location code, Z-order, U-order, X-order, Gray-order

***Abstract*—Space-filling curves have been found useful for many applications in diverse fields. A space-filling curve is a path in a $2^r \times 2^r$ raster domain which visits each location exactly once. In mathematical terms, space-filling curves linearize a two-dimensional integer space, bijectively mapping the space to the integer line.**

An algorithm is presented which generates a large number of space-filling curves/spatial orders. Functions are derived such that the code of each location can be calculated from its coordinates and, conversely, a location code can be decoded to yield the coordinates. The algorithm first generates 4×4 spatial orders which subsequently are scaled up to any desired domain of size $2^r \times 2^r$.

The underlying theory of the algorithm is described in detail as are the processes for scaling up, encoding, and decoding. The curves are generated as a set of incongruent curves, followed, if required, by the sets of associated congruent curves. A number of space-filling curves are illustrated.

I. Literature, History, Motivation, Concepts

Spatial orders and their associated space-filling curves map an n-dimensional space ($n > 1$) to a one-dimensional space (the line of non-negative integers). By means of a space-filling curve, a multi-dimensional problem can be reduced to one dimension, processed in this less-complex state, and then restored to the original dimension if required.

The literature on space-filling curves and spatial orders can be grouped into two classes, applications depending on space-filling curves and papers addressing space-filling curves as such.

Applications can be found in a wide range of areas, concentrating in image and signal processing, parallel and distributed processing, database computing, global optimization, antenna design, and yet others. In image processing, subareas comprise image abstraction, classification, coarsening, compression (both lossless and lossy), encryption and steganography (data and image hiding), multimedia and multispectral processing, and more. Many of the papers include the term 'space-filling curve' in their title, signifying the importance of their role. Examples of articles published in the past five years can be found for all of above areas. Examples: Ouni et al. [1] propose an adaptive algorithm for lossless image compression by exploiting inherent local image coherence by choosing from a fixed set of four space-filling curves. Bhatnagar et al. [2] encrypt images by shuffling the coefficients in a fractional Fourier transform domain according to space-filling curves. Amirtharajan et al. [3] hide confidential data by encrypting them first by a standard method, and then convert the encrypted 1-dimensional data stream into a 2-dimensional pattern using a space-filling curve and hiding that in a cover image. Koga et al. [4] developed an algorithm for image coarsening achieved by generating a minimal length space-filling curve which is adapted to the structure of a given image, preserving its structural context. Lawder et al. [5] discuss multi-dimensional indexing for data-base management systems based on space-filling curves.

The papers on space-filling curves as such deal with properties, valuation metrics and parameters, e.g., domain size. Mokbel et al. [6] propose a metric called *irregularity* which reflects the ordering quality of a given space-filling curve for a large range of dimensions and any space-filling curve. Zarai et al. [7] propose two algorithms, one to calculate the location codes of a Hilbert curve from the locations code of the Z-order, the other from the Gray-order. Chung et al. [8] generalize the domain of Hilbert curves to rectangular domains of arbitrary size, thus removing the size constraint $2^r \times 2^r$. Rather than reducing the dimension, Ahmed et al. [9] utilize the converse situation with a space-filling *surface* algorithm, mapping from 2-d to 3-d, thereby achieving a proximity improvement which is of advantage in parallel computing.

The curves dealt with in above papers are primarily the Hilbert, Peano, Z, Gray, raster scan, and diagonal meander ('zig-zag') space-filling curves, with the Hilbert curve dominating.

Manuscript received January xx, 2014.

Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, Canada V6T 1Z4, schrack@ece.ubc.ca, leos@ece.ubc.ca

New space-filling curves have been proposed previously. Wierum [10] describes the $\beta\Omega$ curve and the H-index curve in the course of assessing a number of curves with the aid of several quality metrics. Liu [11] discusses several new curves; in particular four curves which are related to Hilbert's, comprising, together with a variant of Hilbert's known as Moore's, a complete set. Stocco et al. [12] interleave bits (see [13]) in a systematic fashion, thereby leading to many new space-filling curves; their method is easily extended to three and higher dimensions and to rectangular domains. Haverkort et al. [14] also evaluate a number of space-filling curves from $2^r \times 2^r$ and $3^r \times 3^r$ domains, several of which are new curves.

The history of space-filling curve traces to 1878 when Cantor showed that two finite-dimensional, smooth manifolds are of the same cardinality, even if they are of different dimensions. Specifically, the (closed) unit square $[0,1]\times[0,1]$ can be mapped to the (closed) unit interval $[0,1]$; stated as a corollary, the unit square can be linearized. This is a surprising, non-intuitive result which leads to a large number of practical applications.

Space-filling curves accomplish the linearization of a finite area, volume, or hyper-volume. The first was proposed by Peano in 1890 [15], another by Hilbert a year later [16], Lebesgue in 1904 followed with the Z-curve [17]; yet more were described by other authors in subsequent years. The problems investigated in connection with their research belong to continuous mathematics. For a thorough treatment of this area of mathematics, see Sagan [18].

The motivation for the space-filling curves considered here is an entirely different one. Rather than taking the unit square within the domain $\mathbf{R^2}$ as the range, the space-filling curves of interest are defined on a grid or lattice, i.e. the set of coordinates of ordered integer pairs within the domain $\mathbf{I^2}$; thus these investigations are domiciled in discrete mathematics.

Morton [19] rediscovered Lebesgue's space-filling curve in 1966 in the context of discrete mathematics and computer science. It is now known as the *Morton order* or the *Z-order*. Since then, more space-filling curves were discovered, often restricted to a square domain of size $2^r \times 2^r$, together with a large number of interesting characteristics. Bader [20] describes in depth many of the well-known space-filling curves.

The principal aim of this paper is to present an algorithm and its underlying theory which efficiently generates a large number of spatial orders and the associated space-filling curves in domains of dimensions $2^r \times 2^r$, $r = 2, 3, \cdots$. Here, the terms 'spatial order' and 'space-filling curve' will be used interchangeably since they name the same object in different representations.

Two major subsets can be identified, distinct or *incongruent* and non-distinct or *congruent* space-filling curves; they will be generated separately. Two curves are incongruent if one cannot be derived from the other by a reflection or a rotation. The set of incongruent spatial orders is a proper subset of the congruent set.

A unique label, the *signature* will be associated with each spatial order/space-filling curve. The signature will (a) identify the spatial order, (b) serve as a generating function for the spatial order and establish the order matrix, and (c) define the space-filling curve and facilitate to plot it.

Consider a finite integer grid of size $2^r \times 2^r$, the *domain*, where r is its *resolution*. It is embedded in a coordinate system with coordinate ranges $\{0, 1, 2, \cdots, 2^r - 1\}$. Each coordinate point in the domain defines a *location*. Customarily, the grid is depicted as a square table of 2^{2r} unit squares, each square representing a location.

Starting at some chosen location (the *entry location*), the sequence $0, 1, 2, \cdots, 2^{2r} - 1$ is mapped onto the domain in some fashion, resulting in a square matrix, the *spatial order*. By connecting the locations sequentially, starting at the entry location 0 and ending at location $2^{2r} - 1$ (the *exit location*), a path is generated, the *space-filling curve*.

The domains of space-filling curves can be structured hierarchically. Subdivide a $2^r \times 2^r$ domain recursively: first into four equally-sized square sub-domains, the *quadrants*, of size $2^{r-1} \times 2^{r-1}$, next each of these again, and repeat the process. When the sub-domains have reached dimension 1×1 (the locations), the subdivision process terminates. Following the terminology used for trees (as in data-structures), the domain itself is said to be at *level* r, the first four subdivided sub-domains are each at level $r - 1$, etc.; the locations are at level 0.

One important task related to space-filling curves is to find functions which allow the calculation of the location code n from the two coordinates (*encoding*: $n = f(x, y)$) and the converse operation of finding the coordinates (x, y) of a location from its code (*decoding*: $(x, y) = (g(n), h(n))$). This is possible because the set of coordinates, given by the Cartesian products $\{(x_i, y_i) \,|\, x_i, y_i \in \{0, 1, \cdots, 2^r - 1\}\}$ is mapped bijectively to the location codes, the set $\{0, 1, 2, \cdots, s^{2r} - 1\}$.

II. Combinations and Permutations of Signature Elements

The Domain of Size 2×2

Consider first the domain of size 2×2. The total number of space-filling curves is $4! = 24$, which includes all rotated and reflected curves. To reduce this set to mutually incongruent curves, fix the starting point at the

origin of the coordinate system. The number of space-filling curves reduces to $3! = 6$, but elements of this set still include the reflections on the *major diagonal* (the domain's diagonal passing through the origin). But a curve and its reflection are congruent to each other. Each curve has one such reflection, thus, the number of incongruent curves is $3!/2 = 3$ for a 2×2 domain. They are the *Z-order* (Figure 1), the *U-order* (Figure 2), and the *X-order* (Figure 3) [12], [21], [22], [23], no others exist; therefore, they can be considered to be fundamental and will be referred to as the three *basis orders*.

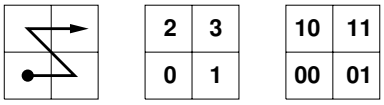


Figure 1: The 2×2 Z-order (y_0, x_0)

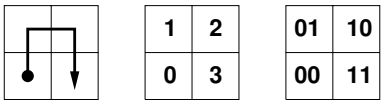


Figure 2: The 2×2 U-order $(x_0, x_0 \oplus y_0)$

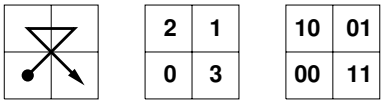


Figure 3: The 2×2 X-order $(x_0 \oplus y_0, x_0)$

The Z-order is of particular importance due to the fact that each location code is obtained by interleaving (see [12]) the binary representation of the coordinates of the location.

Thus, to calculate the location codes of the Z-curve, interleave the bits (the binary digits) of the coordinates; then the four location codes are given by $n(x_0, y_0) = y_0\, x_0$, with $x_0, y_0 \in \{0, 1\}$ (juxtaposition of the bits implies concatenation of the bits). x_0 and y_0 are called the *bit-variables* and for domains of size 2×2, interleaving reduces to juxtaposition of 2 bits. Specifically, the quadruple $(00, 01, 10, 11)_2 = (0, 1, 2, 3)_{10}$ is obtained where the order of the elements in the quadruple is that of the Z-order (Figure 1).

The location codes of the U-curve (Figure 2) are obtained by concatenating the x-coordinate with the function $x \oplus y$: $n(x_0, y_0) = x_0\ x_0 \oplus y_0$, resulting in the quadruple $(0, 3, 1, 2)_{10}$, where the symbol $\oplus$ represents the *bitwise exclusive-OR* (exclusive disjunction) operator. For consistency, plotting a quadruple (in general, a 2^{2r}-tuple for resolutions $r > 0$) requires a convention to be applied to all spatial orders. Since the elements of a quadruple are the location codes at the coordinates from which they were calculated, each must be plotted at its coordinates. The convention adopted here is to order the quadruples (the 2^{2r}-tuples) of coordinates in Z-order. This convention is reflected in Table 1 below. Thus, for the U-order, the quadruple of location codes $(0, 3, 1, 2)_{10}$ maps to the quadruple of coordinate pairs $((0, 0), (1, 0), (0, 1), (1, 1))$, Figure 2.

Similarly, the location codes of the X-curve (Figure 3) are derived from concatenating $x_0 \oplus y_0$ and x_0: $n(x_0, y_0) = x_0 \oplus y_0\ x_0$, the quadruple is $(0, 3, 2, 1)_{10}$.

Thus, the location codes of the three orders are constructed by interleaving the x-coordinate with either the y-coordinate or with the result of the operation $x \oplus y$. Consider the three elements x_0, y_0, and $x_0 \oplus y_0$; choosing pairs, a total of six permutations, $P(3, 2) = 3!/(3-2)! = 6$, viz. $(x_0, y_0), (y_0, x_0), (x_0, x_0 \oplus y_0), (y_0, x_0 \oplus y_0)$, $(x_0 \oplus y_0, x_0), (x_0 \oplus y_0, y_0)$ result. (Recall, for a combination of objects, say 2 from a set of 3, their order is ignored; for a permutation, the order of the objects is essential). The elements are permutable because the operations to interleave and interchange (i.e. permute) are not commutative. Eliminating reflections, the three space-filling curves given by the values $n_Z = y_0\, x_0$, $n_U = x_0\, x_0 \oplus y_0$ and $n_X = x_0 \oplus y_0\, x_0$ remain. The associated ordered pairs $(y_0, x_0), (x_0, x_0 \oplus y_0)$, and $(x_0 \oplus y_0, x_0)$ are the signatures of the basis orders.

The Domain of Size 4×4

Only the basis orders Z, U, and X and their associated congruent orders can be defined on a 2×2 domain ($r = 1$). Since the stated goal is to generate a large number of spatial orders, it is necessary to increase the domain size to the next higher resolution $r = 2$, 4×4 domains, thereby resulting in a significantly larger set of incongruent space-filling curves with a large variety of appearances.

Specifically, for $r = 2$, each coordinate comprises two bits in its binary representation. Let $x = x_1\, x_0$ and $y = y_1\, y_0$, where $x_i, y_i \in \{0, 1\},\ i = 0, 1$ are the bit-variables; the set of bit-variables is $\{x_1, x_0, y_1, y_0\}$, the ranges of the coordinates x and y are $\{0, 1, 2, 3\}_{10}$, and the range of the location codes is $\{0, 1, 2, \cdots, 15\}_{10}$. For the Z-order, the location codes are derived by interleaving its coordinates: $n_Z(x, y) = y_1\, x_1\, y_0\, x_0,\ \ x_i, y_i \in \{0, 1\},\ i = 0, 1$ (again, juxtaposition implies concatenation). For the U-order, $n_U(x, y) = y_1\, x_1 \oplus y_1\, y_0\, x_0 \oplus y_0$; and similarly for the X-order.

There are no reasons, however, to restrict the "assembly" of location codes by interleaving coordinates, *permuting* (i.e. interchanging) their bits also produces space-filling curves. The approach proposed is to permute (a) bits from the binary representation of the coordinates and (b) bits which result from applying exclusive-OR operations to the bits of the coordinates.

The approach here is a generalization of the concept proposed in [12]. It does not produce a complete set of all possible curves in this domain; it does generate, however, the set of curves which can be obtained using only the logical and bit-shifting operations. The set of in-

congruent curves may be augmented by their associated reflections and rotations to provide a total of 322,560 space-filling curves in the 4×4 domain. In Section III, methods are discussed by which the 4×4 domains are scaled up to produce higher-resolution curves.

Since for each space-filling curve there is an entry and an exit location, it is a directed (simple) path. The direction of the path is reversed by interchanging the entry with the exit location and relabelling the locations on the path. The definition of congruency does not take into consideration the direction of a space-filling curve. There are curves for which one of the congruency transformations coincides with its reversed path; for example, a rotation by $180°$ reverses the path of the Z-order (Figure 4). For other spatial orders, none of the congruency transformations coincides with a path reversal.

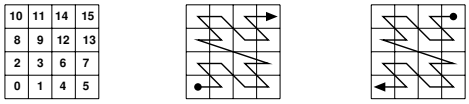


Figure 4: The 4×4 Z-order and Z space-filling curve (y_1, x_1, y_0, x_0) and its $180°$ rotation

Central to the generating algorithm is Table 1 which lists the four bit-variables and the functions obtained by applying the exclusive-OR operator on combinations of two variables $(x_1 \oplus x_0, \cdots)$, three variables $(x_1 \oplus x_0 \oplus y_1, \cdots)$, and four variables $(x_1 \oplus x_0 \oplus y_1 \oplus y_0)$. There are six combinations of two variables, four combinations of three variables, and one for the four variables (permutations cannot be considered since the exclusive-OR operation is commutative). Together with the four variables, there are 15 binary-valued entities to be called the *binary functions*. Thus Table 1 is structured into 15 columns and 16 rows for the 16 locations of 4×4 domains; the rows are ordered according to the convention discussed above, viz. the 4×4 Z-order (Figure 4). Let a location code be represented as a binary integer: $n = n_3\, n_2\, n_1\, n_0,\ n_i \in \{0, 1\},\ i = 0, 1, 2, 3$. Rather than interleaving integer coordinates, different binary functions chosen from Table 1 will be assigned to the four bits of a location code, e.g., $n_3 = x_1 \oplus y_1,\ n_2 = x_1,\ n_1 = x_0 \oplus y_0$, and $n_0 = x_0$. Then $(x_1 \oplus y_1,\ x_1,\ x_0 \oplus y_0,\ x_0)$ is the signature of the spatial order (it is the signature of the 4×4 X-order); the right-most bit is always the least significant bit. The algorithm proceeds in several steps.

Step 1: Generate a preliminary set of quadruples

Select systematically four binary functions from Table 1 by invoking a combinations generator; the resulting set of quadruples is of size $C(15, 4) = 15!/(4!(15-4)!) = 1,365$.

Step 2: Remove invalid quadruples

A closer inspection of the set of quadruples from step 1 reveals that many do not represent a spatial order; they must be eliminated. Consider row 1 of Table 1 for which all entries are zero. Take all combinations of 4 elements from 7, the quadruples from row 1 will evaluate to the zero-quadruple $(0, 0, 0, 0)$.

Lemma. *For each non-zero row of Table 1, 35 combinations can be chosen which are a zero-quadruple; the total for Table 1 is* $15 * 35 = 525$ *zero-quadruples.*

Proof. Each non-zero row has exactly 7 zero elements. The number of zero-quadruples is the combination of four elements chosen from 7, or $C(7, 4) = 7!/(4!(7-3)!) = 35$. There are 15 non-zero rows, thus the total for the table is 525. $\triangle$

col	x1	x0	y1	y0	x1⊕x0	x1⊕y1	x1⊕y0	x0⊕y1	x0⊕y0	y1⊕y0	x1⊕x0⊕y1	x1⊕x0⊕y0	x1⊕y1⊕y0	x0⊕y1⊕y0	x1⊕x0⊕y1⊕y0
row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	1	0	0	1	1	0	1	1	0	1	1
3	0	0	0	1	0	0	1	0	1	1	0	1	1	1	1
4	0	1	0	1	1	0	1	1	0	1	1	0	1	0	0
5	1	0	0	0	1	1	1	0	0	0	1	1	1	0	1
6	1	1	0	0	0	1	1	1	1	0	0	0	1	1	0
7	1	0	0	1	1	1	0	0	1	1	1	0	0	1	0
8	1	1	0	1	0	1	0	1	0	1	0	1	0	0	1
9	0	0	1	0	0	1	0	1	0	1	1	0	1	1	1
10	0	1	1	0	1	1	0	0	1	1	0	1	1	0	0
11	0	0	1	1	0	1	1	1	1	0	1	1	0	0	0
12	0	1	1	1	1	1	1	0	0	0	0	0	0	1	1
13	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0
14	1	1	1	0	0	0	1	0	1	1	1	0	0	0	1
15	1	0	1	1	1	0	0	1	1	0	0	0	1	0	1
16	1	1	1	1	0	0	0	0	0	0	1	1	1	1	0

Table 1: Table of 15 binary functions of 4 variables

When there are two zero-quadruples in a row among rows 2 to 16, it means there are two location codes 0. But it is not possible that two locations with the same code exist in a spatial order; therefore, such a quadruple is called *invalid*. Since there are 525 zero-quadruples in rows 2 to 16, there is a total of $1,365 - 525 = 840$ valid quadruples in the table. The elements of this set will be referred to as the *basis quadruples*.

Lemma. *Only the zero-quadruple can act as the invalidity criterion.*

Proof by counterexample. The quadruple $(0, 0, 0, 1)$ is not present in the set of quadruples of the signature $(x_1, x_0, y_1, x_1 \oplus x_0)$, whereas it is for $(x_1, y_1, x_1 \oplus y_1, x_0 \oplus y_1)$, even though both are invalid. Examples of pairs of signatures can be found for all other non-zero quadruples where for one signature, a zero-quadruple is present but not for the other. When the zero-quadruple appears twice in a set of quadruples, it must replace

one of the non-zero quadruples but which one it is depends entirely on the binary functions of the associated quadruple; thus, non-zero quadruples do not qualify as the invalidity/elimination criterion. △

Step 3: Expand the set of basis quadruples

By permuting the elements of a basis quadruple, more space-filling curves are generated.

Lemma. *Permuting the elements of a signature does not invalidate the resulting signature.*

Proof. The signature of a spatial order represents the 16 (numerical) quadruples obtained by evaluating each of the binary functions of the signature at each location of the domain. A signature is invalid when there is more than one zero-quadruple among the 16 quadruples. Permuting the elements of a quadruple will not change any of the values, only their order is changed. Thus, no additional zero-quadruples are introduced and the permuted signature is also a signature (a valid quadruple).△

For each basis quadruple there are $P(4,4) = 4! = 24$ spatial orders for a total of $840 * 24 = 20,160$ signatures.

Step 4: Eliminate reflections on the major diagonal

The set from step 3, however, includes congruent pairs: a curve and its reflection on the major diagonal (reflection is a commutative operation). By definition, a geometric object is *congruent* to another geometric object if the two coincide after one or more of the transformations *translation, scaling, rotation*, and *reflection* has been applied. Space-filling curves are geometric objects for which the transformations translation and scaling need not be considered. There remain rotations by $0°$, $90°$, $180°$, and $270°$, reflections about the major diagonal, the *minor diagonal* (orthogonal to the major diagonal), the horizontal axis, and the vertical axis (all reflection axes pass through the centre of the domain); in total 8 curves.

The reflection of a curve on the major diagonal is obtained by interchanging the coordinate axes, i.e. $y_i \rightarrow x_i$, and $x_i \rightarrow y_i$, $i = 0, 1$ (the 'substitution rule'). For example, the two signatures $(x_1, x_1 \oplus y_1 \oplus y_0, x_0, x_0 \oplus y_1)$ (Figure 5) and $(y_1, x_1 \oplus x_0 \oplus y_1, y_0, x_1 \oplus y_0)$ (Figure 6) represent such a pair, hence the spatial orders they represent are congruent. After elimination, there remain $20160/2 = 10,080$ quadruples which are the signatures of a set of incongruent curves.

Lemma. *The binary functions of Table 1 can be paired according to the substitution rule $x_i \leftrightarrow y_i$, $i = 0, 1$. They will be called* reflection pairs*. There are three exceptions of functions which are self-reflecting.*

Proof. Consider a binary function and apply the substitution rule, resulting in a reflection pair. Substitution is a commutative operation, hence a reflection pair is mutually reflective. For example, x_1 and y_1 are a reflection pair, as are $x_1 \oplus x_0 \oplus y_1$ and $y_1 \oplus y_0 \oplus x_1 = x_1 \oplus y_1 \oplus y_0$. The self-reflecting functions are $x_1 \oplus y_1$, $x_0 \oplus y_0$, and $x_1 \oplus x_0 \oplus y_1 \oplus y_0$ (exclusive OR is a commutative operation). △

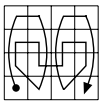


Fig. 5

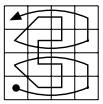


Fig. 6

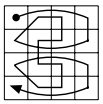


Fig. 7

Figure 5: A 4×4 space-filling curve from the U-order family; $(x_1, x_1 \oplus y_1 \oplus y_0, x_0, x_0 \oplus y_1)$

Figure 6: Figure 5 reflected on the major diagonal $(y_1, x_1 \oplus x_0 \oplus y_1, y_0, x_1 \oplus y_0)$

Figure 7: Figure 5 rotated by $270°$ or the path of Figure 6 reversed; $(\overline{y_1}, \overline{x_1 \oplus x_0 \oplus y_1}, \overline{y_0}, \overline{x_1 \oplus y_0})$

The reflection of a quadruple is easily generated with the aid of a look-up table. Assign to each binary function an index number (the column number in Table 1); the look-up table becomes $T_r = [3, 4, 1, 2, 10, 6, 8, 7, 9, 5, 13, 14, 11, 12, 15]$; i.e. $(i, T_r(i)), i = 1, \cdots, 15$ are reflection pairs. Let (s_3, s_2, s_1, s_0) be a given signature and (r_3, r_2, r_1, r_0) the signature of its reflection, where $r_i = T_r(s_i)$, $i = 0, 1, 2, 3$. For most quadruples generated by step 3, either $s_3 < r_3$ or $r_3 < s_3$. Thus, to eliminate the reflection of a quadruple, generate its reflection, then accept the quadruple if $s_3 < r_3$. If the quadruple has been accepted, its reflection, which will be generated later, will be rejected. Conversely, if it has not been accepted, the reflection will be accepted later. There are cases when $s_3 = r_3$, which will happen when s_3 is one of the self-reflecting binary functions. For these, the acceptance criterion is $s_2 < r_2$, unless $s_2 = r_2$, in which case the criterion is $s_1 < r_1$. The case $s_1 = r_1$ will not occur because all quadruples which have the three self-reflecting functions as elements are invalid.

Step 5: Expand the set of incongruent quadruples

The rotation and reflection transformations (except the reflection on the major diagonal) are implemented by inverting combinations of the bit-variables. (A bit-variable, the value of which is a single bit, is inverted by applying the *bitwise NOT* operator $\overline{x}$, i.e. by a one's complement operation.)

Inverting none, one, two, three, or all four bit-variables produces 16 signatures. Applying the same operations to the reflections on the major diagonal doubles that number to 32. But there are only eight congruent curves for a given curve, implying that by introducing inversions, three additional incongruent curves are generated, to a sum of 4. The total becomes $10080 * 4 = 40,320$ space-filling curves, none of which is congruent to any other in that set.

Lemma. *Inverting x_0 or y_0 or both x_0 and y_0 produces spatial orders which are incongruent to the spatial*

orders generated by step 4.

Proof. Inverting x_0 interchanges row 0 with row 1 and row 2 with 3. The location code at coordinates $(0,0)$ is moved to $(1,0)$. But the entry location of every spatial order generated by step 4 is at $(0,0)$, furthermore, congruence transformations move it to one of the four domain vertices at $(0,0),(3,0),(0,3)$ or $(3,3)$. Thus a spatial order with an entry location at coordinates of $(1,0),(0,1)$ or $(1,1)$ is incongruent to the ones from step 4. Similar arguments apply to inverting y_0 or both x_0 and y_0. $\triangle$

Thus, inverting x_0 results in space-filling curves with the entry location at (1,0), inverting y_0 at $(0,1)$ and inverting both at (1,1) (Figure 8d).

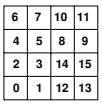


Fig. 8a

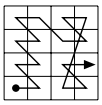


Fig. 8b

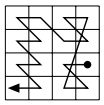


Fig. 8c

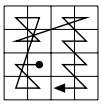


Fig. 8d

Figure 8: The hybrid 4×4 ZU space-filling curves $(x_1, x_1 \oplus y_1, y_0, x_0)$, $(\overline{x_1}, \overline{x_1 \oplus y_1}, \overline{y_0}, \overline{x_0})$ and $(x_1, x_1 \oplus y_1, \overline{y_0}, \overline{x_0})$

Step 6: Generate the signatures of congruent spatial orders

A reflection on the major diagonal cannot be implemented using inversions. Therefore, the spatial orders which are reflections on the major diagonal must be present in the set at this point, either by retaining them by skipping step 4 above or by regenerating them, e.g., by a mapping operation using a table look-up. Ignoring this consideration will fail to include orders reflected on the major and minor diagonals and those rotated by $90°$ and $270°$.

Lemma. *Congruence transformations are obtained by inverting certain combinations of the bit-variables x_1, x_0, y_1, y_0 as follows.*

Reflection on the vertical domain axis: invert x_1 and x_0; reflection on the horizontal domain axis: invert y_1 and y_0; rotation by $180°$: invert x_1, x_0, y_1, y_0; reflection on the major diagonal: substitute $x_i \leftrightarrow y_i,\ i = 0, 1$; rotation by $90°$: invert x_1, x_0 followed by the substitution $x_i \leftrightarrow y_i,\ i = 0, 1$; rotation by $270°$: invert y_1, y_0 followed by the substitution $x_i \leftrightarrow y_i,\ i = 0, 1$; reflection on the minor diagonal: invert x_1, x_0, y_1, y_0 followed by the substitution $x_i \leftrightarrow y_i,\ i = 0, 1$.

Proof. Inverting x_0 (the least significant bit of the x-coordinate values) interchanges columns of the domain: column 0 with column 1 and column 2 with 3; inverting x_1 interchanges columns 0 and 1 with columns 2 and 3; inverting both x_0 and x_1 (y_0 and y_1) results in the reflection of the domain on the vertical (horizontal) axis; carrying out both the vertical and horizontal reflections then results in the rotation by $180°$. Substituting $x_i \leftrightarrow y_i,\ i = 0, 1$ interchanges the coordinate axes, i.e. reflects on the major diagonal; rotation by $90°$ ($270°$) is the result of reflecting on the vertical (horizontal) axis followed by reflecting on the major diagonal; reflection on the minor diagonal is achieved by rotating by $180°$ followed by reflecting on the major diagonal. $\triangle$

Inverting combinations of the bit-variables of a signature in the set for all permutations and the permutations of its reflection yields eight for each signature, the set of congruent curves. The total score becomes $40320 * 8 = 322,560$ space-filling curves. Compared to the number of all possible curves of a 4×4 domain, $16! \approx 2.09\,10^{13}$, it is still a small set.

Since with each incongruent curve there are 8 associated congruent curves (including itself), the total of incongruent curves in the 4×4 domain is $16!/8 \approx 2.6\,10^{12}$. Compared to the set of generated incongruent curves of size 40,320, that set is also small.

Step 7: Scale up the 4×4 domain (as discussed in Section III)

Step 8: Generate the spatial order and its associated curve from a signature

The final step generates the spatial order and curve from its signature. For $r = 2$, the spatial order is a 4×4 matrix of location codes, arranged according to their coordinates. Each location code is derived from the signature by substituting the bits of the coordinates into the elements of the signature, evaluating them and concatenating the resulting four bits.

To plot the space-filling curve, start at the location with code 0 and draw the path by visiting the locations in the sequence $0, 1, 2, \cdots, 15$. Figures 8a and 8b are an example of a spatial order and its associated curve.

The signature of a space-filling curve with its path reversed is obtained by inverting each element of its signature. For example, the signature $(\overline{x_1}, \overline{x_1 \oplus y_1}, \overline{y_0}, \overline{x_0})$ represents the curve which is identical to that of Figure 8b except that its path is reversed (Figure 8c, compare also Figures 6 and 7 and their signatures). Figure 8 also demonstrates that a curve with reversed path need not coincide with any of its congruent curves.

Additional considerations. Using appropriate programming code, the algorithm can be tailored to the specific requirements that a project/application may call for. Thus, a variety of sets, differing in size or in sequence, or sets comprising spatial orders with a common characteristic, can be accommodated. For example, sets can be reduced in size by skipping one or more of steps 3, 4, 5, or 6, or by selecting only every *m*th signature, or by replacing Table 1 by a smaller one where some of the binary functions were omitted.

The sequence of generation can be altered by rearranging the binary functions in Table 1 or by changing the combinations generator in step 1 or the permutation

generator in step 3. Two complementary sets can be obtained by generating one set first with the inequality $s_i < r_i$ in step 4 and then again with the inequality inverted to $s_i > r_i$. A set with spatial orders which have one common entry location (any one of the 16 locations in the domain) or spatial orders where all have the same orientation can be produced easily.

If only one or a very few spatial orders need to be generated, first, pick four binary functions from the set of 15 functions listed in Table 1 to form a preliminary quadruple. Second, determine its validity by evaluating the four elements of the quadruple for all coordinates. If the quadruple evaluates to $(0, 0, 0, 0)$ no more than once for the 16 evaluations, a 4×4 spatial order has been found. Third, if required, invert one or more of the bit-variables in the signature according to the desired transformation (rotation, reflection).

III. Generalisations to domains of resolution $r > 2$

For most applications, domains of resolution $r = 1$ and $r = 2$ are insufficient in size. It is quite simple, fortunately, to increase the resolution. Several approaches are available: (a) replicate a domain recursively, (b) replicate a domain fourfold and connect the four replicates as given by another spatial order of the resolution of the original domain, (c) generate the signature of a spatial order by combining the signatures of two or more lower-resolution spatial orders, (d) generalize the algorithm introduced above.

Approach (a) is the classical approach. Consider one of the basis orders or one of their congruent orders, replicate it four times (becoming four sub-domains) and arrange these in a square. Connect the four sub-domains by the order of the original curve, resulting in a 4×4 domain. Iterate the process — the second iteration generates an 8×8 domain; repeat until the desired resolution/domain size is reached (Figures 9 and 10).

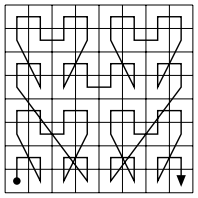


Fig. 9

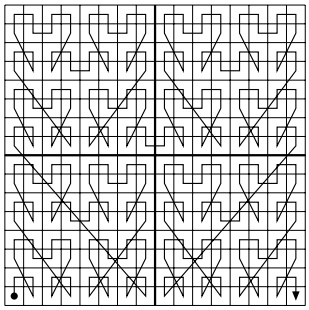


Fig. 10

Figure 9: The 8×8 U-order space-filling curve $(x_2, x_2 \oplus y_2, x_1, x_1 \oplus y_1, x_0, x_0 \oplus y_0)$

Figure 10: The 16×16 U-order space-filling curve $(x_3, x_3 \oplus y_3, x_2, x_2 \oplus y_2, x_1, x_1 \oplus y_1, x_0, x_0 \oplus y_0)$

An order from the set of congruent orders that is not a 4×4 Z-, U-, or X-order, however, must be replicated 16 times to allow connecting the sub-domains according to the original curve, resulting in a 16×16 domain on the first iteration (Figures 11 and 12).

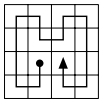


Fig. 11

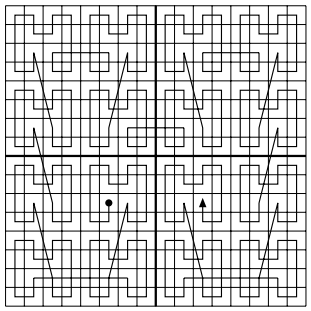


Fig. 12

Figure 11: The 4×4 H-order space-filling curve $(x_1, x_1 \oplus y_1, \overline{x_0} \oplus y_1, \overline{x_0} \oplus \overline{y_0})$ [23]

Figure 12: The 16×16 H-order $(x_3, x_3 \oplus y_3, \overline{x_2} \oplus y_3, \overline{x_2} \oplus \overline{y_2}, x_1, x_1 \oplus y_1, \overline{x_0} \oplus y_1, \overline{x_0} \oplus \overline{y_0})$

Approach (b) exploits the observation that there is no compelling reason to employ the same spatial order to connect a set of sub-domains at a certain level to obtain a higher-level domain. To see this, consider the 2×2 Z-curve, and replace each location by, say, a 2×2 X-curve, resulting in a 4×4 spatial order of four 2×2 X-curves connected by the Z-curve (Figure 13, see also Figure 8).

Similarly, replicating an order from the set of congruent orders four times and connecting the sub-quadrants in turn by one of the orders from the set of 2×2 basis orders results in an 8×8 domain (Figures 14 and 15).

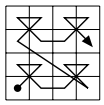


Fig. 13

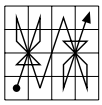


Fig. 14

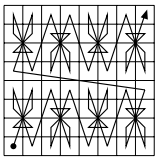


Fig. 15

Figure 13: A 4×4 hybrid XZ curve $(y_1, x_1, x_0 \oplus y_0, x_0)$

Figure 14: The 4×4 space-filling curve $(x_1, x_1 \oplus x_0 \oplus y_1, x_1 \oplus x_0 \oplus y_0, x_0 \oplus y_1 \oplus y_0)$

Figure 15: Figure 14 scaled up to size 8×8 $(y_2, x_2, x_1, x_1 \oplus x_0 \oplus y_1, x_1 \oplus x_0 \oplus y_0, x_0 \oplus y_1 \oplus y_0)$

Approach (c) is the algebraic version of approach (b). The signature $(x_0, x_1 \oplus y_1, x_1, x_0 \oplus y_0)$ represents the curve shown in Figure 16. Prepend this signature by (y_3, y_2, x_3, x_2), the indices-adjusted signature of the raster-scan order (Figure 17) for level $r = 3$. The result is the signature $(y_3, y_2, x_3, x_2, x_0, x_1 \oplus y_1, x_1, x_0 \oplus y_0)$ of the space-filling curve depicted in Figure 18.

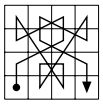


Fig. 16

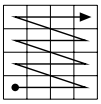


Fig. 17

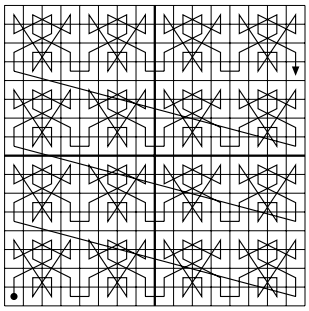


Fig. 18

Figure 16: The Bat space-filling curve $(x_0, x_1 \oplus y_1, x_1, x_0 \oplus y_0)$

Figure 17: The 4×4 Raster-scan space-filling curve (y_1, y_0, x_1, x_0)

Figure 18: A hybrid space-filling curve by combining Figures 14 and 15; $(y_3, y_2, x_3, x_2, x_0, x_1 \oplus y_1, x_1, x_0 \oplus y_0)$

Approach (d) utilizes the principles of the proposed algorithm. For the resolution $r = 3$ (8×8 domains), for example, there are six bit-variables and 63 binary functions. Choose six functions from the set of binary functions. Determine the validity of the selected 6-tuple; if valid, the 6-tuple is the signature; else obtain another selection; example: Figure 19.

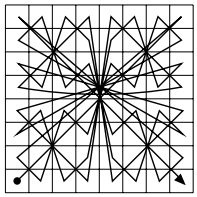


Figure 19: The Snowflake space-filling curve $(x_2 \oplus y_2, x_1 \oplus y_1, x_0 \oplus y_0, x_2, x_1, x_0)$ [12]

Approaches (a) to (d) do not exhaust the possibilities to increase the resolution.

IV. Encoding and Decoding

The ability to encode and decode all spatial orders is considered a major characteristic and advantage of the described algorithm.

To encode and decode spatial orders, the signature must be known. Encoding is straight-forward: substitute the bits of a coordinate into the signature, resulting in the location code at that location.

There are two methods for decoding. The arithmetic method needs to be implemented in stages with the aid of Table 1. Separate the four least significant bits of the given location code and rearrange the four bits according to the order of the binary functions of Table 1. If the signature is stored as a tuple of indices (i.e. the column numbers shown in Table 1), the rearrangement is achieved by sorting the binary function indices in ascending order and rearrange the location code bits accordingly. Example: the signature of the order of Figure 16 is $(x_0, x_1 \oplus y_1, x_1, x_0 \oplus y_0)$ and is represented by the index-tuple $(2, 6, 1, 9)$. Sorted in the order of Table 1, it becomes $(1, 2, 6, 9)$. The location code $n = 13_{10} = 1101_2$, rearranged as was the signature, becomes 0111_2. By searching Table 1 under columns (1,2,6,9) for 0111_2, row 10 is found for which $(x, y) = (1, 2)$, the expected result. For orders $r > 2$, repeat the process by matching the next four least significant bits of the location code to yield x_3, x_2, y_3, y_2; iterate as required for higher levels. Finally, the coordinate bits need to be concatenated to obtain the coordinates.

Alternatively, an algebraic formula approach may be taken to decode a location code. From the signature $s = (s_3, s_2, s_1, s_0)$, where s_i is one of the 15 binary functions, it is easy to derive the four formulas for the bits of the coordinates: $x_1 = f_1(s_3, s_2, s_1, s_0)$, $x_0 = f_2(s_3, s_2, s_1, s_0)$, and similarly for y_1, y_0, by using the exclusive-OR identity $y \equiv (x \oplus y) \oplus x$.

For the previous example, the four coordinate functions become $x_1 = s_1, x_0 = s_3, y_1 = s_2 \oplus s_1, y_0 = s_3 \oplus s_0$, from which $x_1 = n_1, x_0 = n_3, y_1 = n_2 \oplus n_1, y_0 = n_3 \oplus n_0$, where the n_i are again the location code bits. Thus, $n = 13$ yields $x_1 = 0, x_0 = 1, y_1 = 1 \oplus 0 = 1, y_0 = 1 \oplus 1 = 0$ or $(x, y) = (1, 2)$, as before.

The advantage of the arithmetic method is that it can be programmed for spatial orders generated internally whereas for the algebraic approach the coordinate functions need to be derived and programmed explicitly for each order.

V. Visual Assessments

Since the set of incongruent orders is so large, a visual inspection of all its elements was not attempted; only a few are presented here (of size 4×4), chosen at random.

Among the curves in the set of incongruent orders are the Z-, U-, and X-orders as well as others related to them, which frequently is visually obvious. For example, the orders shown in Figures 8, 20, and 21 are clearly related to the Z-order, and the curves shown in Figures 5, 11, 22, 23, and 24 are part of the U-family; similarly, the curves of Figures 25, 26, and 27 are from the X-family. Yet others cannot be clearly associated with the Z-, U-, or X-order family (Figures 14, 28, 29, 30, and 31).

Some well-known curves are in that set, e.g., the raster-scan (Figure 17), interlaced raster-scan, meander scan (row prime scan), and the curves generated by Gray codes [24], [25], [27], (e.g., Figures 22 and 23). The Hilbert curve [26], however, requires a two-part signature for its definition and therefore is not generated.

To demonstrate the effect of permuting signature elements, compare Figure 25 with 28 and 32: the sets of

signature elements are identical, but not their order, and the three curves differ substantially in appearance.

A space-filling curve, reflected on its major diagonal followed by inverting y_1 and y_0 is rotated by 270° (90° clockwise) (Figures 25 and 26); by inverting x_0 and y_0, a curve with entry location at $(1, 1)$ results (Figure 27).

Changing a single variable in a signature by substituting a variable with another one (e.g., by changing the index of one of the variables) may render the quadruple invalid. If it is not invalid, the change will produce an order which may be markedly different from the original (Figures 20 and 33).

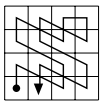


Fig. 20

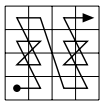


Fig. 21

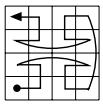


Fig. 22

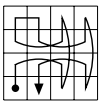


Fig. 23

Figure 20: The signature of Figure 25 with change of one index; $(x_0, x_1 \oplus y_1, x_1 \oplus y_0, x_0 \oplus y_0)$
Figure 21: The 4×4 Double-Z space-filling curve (x_1, y_0, y_1, x_0)
Figure 22: A Gray-order space-filling curve $(y_1, x_1 \oplus y_1, x_1 \oplus y_0, x_0 \oplus y_0)$
Figure 23: An anti-Gray-order space-filling curve $(x_0, x_0 \oplus y_1, x_1 \oplus x_0 \oplus y_1, x_1 \oplus x_0 \oplus y_1 \oplus y_0)$ [27], [23]

Generally, as a cursory overview shows, some space-filling curves are quite distinctive and, arguably, visually pleasing. Many possess a vertical, horizontal or other symmetry.

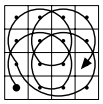


Fig. 24

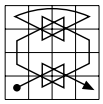


Fig. 25

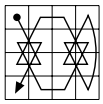


Fig. 26

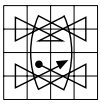


Fig. 27

Figure 24: The Rose space-filling curve $(y_0, x_0, x_1, x_1 \oplus y_1)$
Figure 25: The space-filling curve $(x_0, x_0 \oplus y_1, x_1 \oplus y_0, x_0 \oplus y_0)$
Figure 26: Figure 25 rotated 270° $(\overline{y_0}, x_1 \oplus \overline{y_0}, x_0 \oplus \overline{y_1}, x_0 \oplus \overline{y_0})$
Figure 27: Inverted x_0 and y_0 in the signature of Figure 25; $(\overline{x_0}, \overline{x_0} \oplus y_1, x_1 \oplus \overline{y_0}, \overline{x_0} \oplus \overline{y_0})$

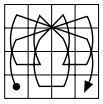


Fig. 28

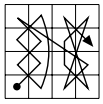


Fig. 29

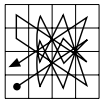


Fig. 30

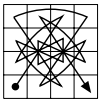


Fig. 31

Figure 28: The Spider space-filling curve, a permutation of the signature of Figure 25 $(x_1 \oplus y_0, x_0 \oplus y_0, x_0, x_0 \oplus y_1)$
Figure 29: The space-filling curve $(x_1, x_0 \oplus y_0, y_1, x_1 \oplus y_0)$
Figure 30: The space-filling curve $(x_1 \oplus x_0 \oplus y_0, x_1 \oplus y_1 \oplus y_0, x_0 \oplus y_1 \oplus y_0, x_1 \oplus x_0 \oplus y_1 \oplus y_0)$
Figure 31: The space-filling curve $(x_0, x_1 \oplus y_1, x_1 \oplus y_0, x_0 \oplus y_1)$

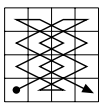


Fig. 32

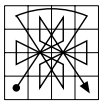


Fig. 33

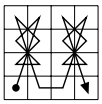


Fig. 34

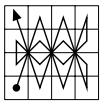


Fig. 35

Figure 32: Another permutation of the signature of Figure 25; $(x_1 \oplus y_0, x_0, x_0 \oplus y_1, x_0 \oplus y_0)$
Figure 33: The signature of Figure 25 with change of one variable; $(x_0, x_0 \oplus y_1, x_1 \oplus x_0, x_0 \oplus y_0)$
Figure 34: The space-filling curve $(x_1, x_0 \oplus y_0, x_0 \oplus y_1, x_0)$
Figure 35: The space-filling curve $(x_0 \oplus y_1, x_0 \oplus y_0, x_1 \oplus x_0 \oplus y_0, y_0)$

Finally, two orders are shown which cannot be generated by the algorithm presented (Figures 38 and 39). The signature of the spiral curve cannot be represented by the binary functions of Table 1 with the exception of s_0 ($s_0 = x_0 \oplus y_0$). The signature of the curve of Figure 39 requires a two-part signature, $s = (x_1, x_1 \oplus y_2, x_0 \oplus y_0, s_0)$, where $s_0 = x_1 \oplus y_0$ when $y_1 = 0$ and $s_0 = x_0$ when $y_1 = 1$.

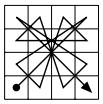


Fig. 36

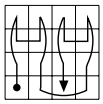


Fig. 37

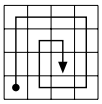


Fig. 38

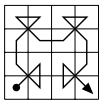


Fig. 39

Figure 36: The Samurai space-filling curve $(x_1 \oplus y_1, x_0 \oplus y_0, x_1, x_0)$ [12]
Figure 37: The space-filling curve $(x_1, x_1 \oplus x_0, x_1 \oplus x_0 \oplus y_1, x_1 \oplus x_0 \oplus y_1 \oplus y_0)$
Figure 38: A spiral space-filling curve
Figure 39: A space-filling curve from the X-family

VI. Conclusion

An algorithm is presented which generates large sets of spatial orders/space-filling curves which are either incongruent to each other or, if desired, include all congruent orders. The generation can be altered to include subsets of any size. A spatial order is definable by three representations: (a) graphically, as illustrated with examples, (b) numerically/arithmetically by a matrix of the location codes, and (c) algebraically by a $2\,r$-tuple (where r is the resolution of the spatial order) called its signature. A signature is a generating function, it enables the encoding of the coordinates, i.e. calculating the location code. A signature may also be utilized as a unique label for the order it generates. The converse operation, decoding or extracting the coordinates from a given location code, is just as straightforward; decoding is accomplished either numerically or by decoding functions that can be derived from the signature.

Only three incongruent spatial orders, the Z-, U-, and X orders, can be defined on the 2×2 domain (the

domain of resolution 1), whereas the 4×4 domain (resolution 2) supports a large number of spatial orders. Thus, the algorithm generates spatial orders for domains of resolution 2 which then can be scaled up to resolutions greater than 2 by several techniques.

The algorithm is based on treating the bits of the coordinates as variables; the binary functions of the signature are defined with the exclusive-OR and the bitwise-NOT operators. From the four variables x_0, x_1, y_0, y_1 (the bits of the coordinates), 15 binary functions can be defined and organized in a 16×15 table. Selecting four functions from the 15 yields a quadruple which may or may not define a spatial order; eliminating invalid quadruples is, however, quite simple with the aid of the function table. Selecting all combinations of quadruples and eliminating the invalid ones results in a set of 10,080 incongruent spatial orders. This set can be increased four-fold to a larger set of incongruent spatial orders by inverting one or both x_0, y_0. Finally, all congruent spatial orders are obtained by inverting all combinations of coordinate bits, resulting in a set of size 322,560.

The algorithm described has been encoded and tested using MATLAB [28], verifying the results.

A survey of some spatial orders from the set of incongruent orders reveals that the Z-, U-, and X-orders of resolution 2 are members, as are other well-known ones, e.g., the raster-scan, meander- and Gray orders. Examples of selected spatial orders are included.

A visual assessment of some of the space-filling curves of that set shows spatial orders which exhibit much symmetry and are frequently of pleasing appearance; others seem to have been generated, counter-intuitively, at random without discernible order.

A generalization of the algorithm to higher dimensions, e.g., the $4 \times 4 \times 4$ voxel domain, is straight-forward by including a third variable z.

REFERENCES

[1] T. Ouni, A. Lassoued, and M. Abid, "Adapted Scan Based Lossless Image Compression," in *SIP 2011*, *CCIS 260*, pp. 82–91, 2011.

[2] G. Bhatnagar, Q.M.J. Wu, and B. Raman, "Image and Video Encryption based on Dual Space-Filling Curves," *The Computer Journal*, vol. 55, no. 6, pp. 667–685, 2012.

[3] R. Amirtharajan and J.B.B. Rayappan, "An intelligent chaotic embedding approach to enhance stego-image quality," *Information Sciences*, vol. 193, pp. 115–124, 2012.

[4] T. Koga and N. Suetake, "Image coarsening by using space-filling curve for decomposition-based image enhancement," *J. Vis. Commun. Image R.*, vol. 24, pp. 806–818, 2013.

[5] J.K. Lawder and P.J.H. King, "Using Space-Filling Curves for Multi-dimensional Indexing," *BNCOD 17, Lecture Notes in Computer Science*, vol. 1832, pp. 20–35, 2000.

[6] M.F. Mokbel and W.G. Aref, "Irregularity in high-dimensional space-filling curves," *Distributed and Parallel Databases*, vol. 29, pp. 217–238, 2011.

[7] Y. Zarai and S. Rakib, "Hilbert Space-Filling by Piecewise-Linear Index Transformation," *IEEE Signal Processing Letters*, vol. 15, pp. 717–720, 2008.

[8] K.-L. Chung, Y.-Y. Huang, and Y.-W. Liu, "Efficient algorithms for coding Hilbert curve of arbitrary-sized image and application to window query," *Information Sciences*, vol. 177, pp. 2130–2151, 2007.

[9] M. Ahmed and S. Bokhari, "Mapping with Space Filling Surfaces," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 9, pp. 1258–1269, 2007.

[10] J.-M. Wierum, "Logarithmic Path-Length in Space-Filling Curves," in *Canadian Conference on Computational Geometry*, pp. 22–26, 2002.

[11] X. Liu, "Four alternative patterns of the Hilbert curve," *Applied Mathematics and Computation*, vol. 147, pp. 741–752, 2004.

[12] L.J. Stocco and G. Schrack, "On spatial orders and location codes," *IEEE Transactions on Computers*, vol. 58, no. 3, pp. 424–432, 2009.

[13] L. Stocco and G. Schrack, "Integer Dilation and Contraction for Quadtrees and Octrees," in *Proc. IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing (PACRIM '95)*, Victoria, BC, Canada, pp. 426–428, 1995.

[14] H. Haverkort, and F. van Walderveen, "Locality and bounding-box quality of two-dimensional space-filling curves," *Computational Geometry*, vol. 43, pp. 131–147, 2010.

[15] G. Peano, "Sur une courbe, qui remplit toute une aire plane," *Math. Ann.*, vol. 36, pp. 157–160, 1890.

[16] D. Hilbert, "Über die stetige Abbildung einer Linie auf ein Flächenstück," *Math. Ann.*, vol. 38, pp. 459–460, 1891.

[17] H.L. Lebesgue, *Leçons sur l'intégration et la recherche des fonctions primitives*, Paris: Gauthier-Villars, pp. 44–45, 1904.

[18] H. Sagan, *Space-Filling Curves*, New York, Berlin, etc.: Springer, 1994.

[19] G.M. Morton, "A computer oriented geodetic data base; and a new technique in file sequencing," *Technical Report*, Ottawa, Canada: IBM Ltd., 1966.

[20] M. Bader, *Space-Filling Curves, An Introduction with Applications in Scientific Computing*, Berlin, Heidelberg, etc.: Springer-Verlag, 278 pp., 2013.

[21] X. Liu and G.F. Schrack, "A New Ordering Strategy Applied to Spatial Data Processing," *International Journal of Geographical Information Science*, vol. 12, no. 1, pp. 3–22, 1998.

[22] G. Schrack and X. Liu, "The Spatial U-Order and Some of its Mathematical Characteristics," in *Proc. IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing (PACRIM '95)*, Victoria, BC, Canada, pp. 416–419, 1995.

[23] X. Liu, "On the Ordering of Multiattribute Data in Information Retrieval Systems," *Doctoral Thesis, The University of British Columbia (Canada)*, 1996, https://cIRcle.ubc.ca/handle/2429/4787

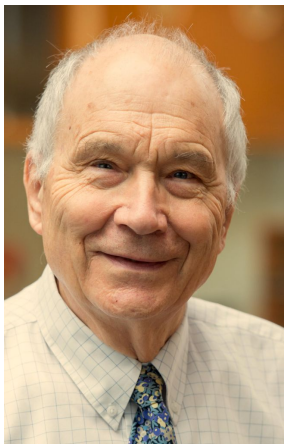
[24] C. Savage, "A survey of combinatorial Gray codes," *SIAM Review*, vol. 39, pp. 55–63, 1997.

[25] X. Liu and G.F. Schrack, "A heuristic approach for constructing symmetric Gray codes," *Applied Mathematics and Computation*, vol. 155, pp. 55–63, 2004.

[26] X. Liu and G. Schrack, "Encoding and decoding the Hilbert order," *Software—Practice and Experience*, vol. 26, no. 12, pp. 1335–1346, 1996.

[27] R. Hamming, *Coding and Information Theory (2nd Ed.)*, Englewood Cliffs, NJ: Prentice-Hall, 1986.

[28] www.mathworks.com



Günther Schrack received a Dr. Math. degree from the Eidgenössische Technische Hochschule (ETH, Swiss Federal Institute of Technology) in Zürich in 1968 in Applied Mathematics. Prior to that he received a Bachelor of Applied Science (BASc) and Master of Applied Science (MASc) in Electrical Engineering from the University of British Columbia, Vancouver, Canada. Dr. Schrack is now Professor Emeritus of Computer Engineering at the University of British Columbia.

In the past, Dr. Schrack was active in numerical optimization, in computer graphics, particularly in high-level programming languages for computer graphics, and in the correlation between graphics and music; at the present time, his research interests concentrate on space-filling curves and spatial orders and their characteristics.

Leo Stocco received his PhD in Electrical Engineering at the University of British Columbia in 2000. He joined Integrated Surgical Systems to lead the development of the ROBODOC surgical assistant before returning to UBC in 2005 as a faculty member. He was awarded the Killam Award for outstanding teaching in 2012.

His interests include gears and gear pumps, robotics, computer graphics and electro-mechanical rapid prototyping.