Hilbert Order in d Dimensions

June 19, 2000

June 2000

The author gratefully acknowledges support from the taxpayers of Québec and of Canada who have paid his salary and research grants while this work was developed at McGill University, and from his students (who built the implementations and investigated the data structures and algorithms) and their funding agencies.

Hilbert curves are one-dimensional orderings of the points of a rectilinear grid in d dimensions. They have the property that every nearest neighbour on the curve is also a nearest neighbour in d-space. In some sense they thus maximally preserve clustering from d-space to 1-space. They are defined recursively, starting from a base curve on the d-hypercube; each grid point is then replaced by a d-hypercube at finer resolution, with its vertices connected by the base curve rotated or reflected suitably. These are *spacefilling* curves. [4, 2]

There are many possible such orderings. We somewhat arbitrarily pick one and call it "the Hilbert curve". One restriction we impose is to use only reflections in the transformation.

1 The Base Curve

Another restriction is to take as the base curve a "binary reflected Gray code". Gray codes[1] are sequences of bit codes in which each code differs from ite predecessor in only one bit. Such a code with d bits corresponds to a sequence of nearest neighbours in a d-hypercube. A d-bit binary reflected Gray code is built recursively from the $(d-1)$-bit code by putting a 0 before each code in the sequence, then following this with the reversal of the sequence with a 1 before each code. Here are the sequences for $d = 1..4$.

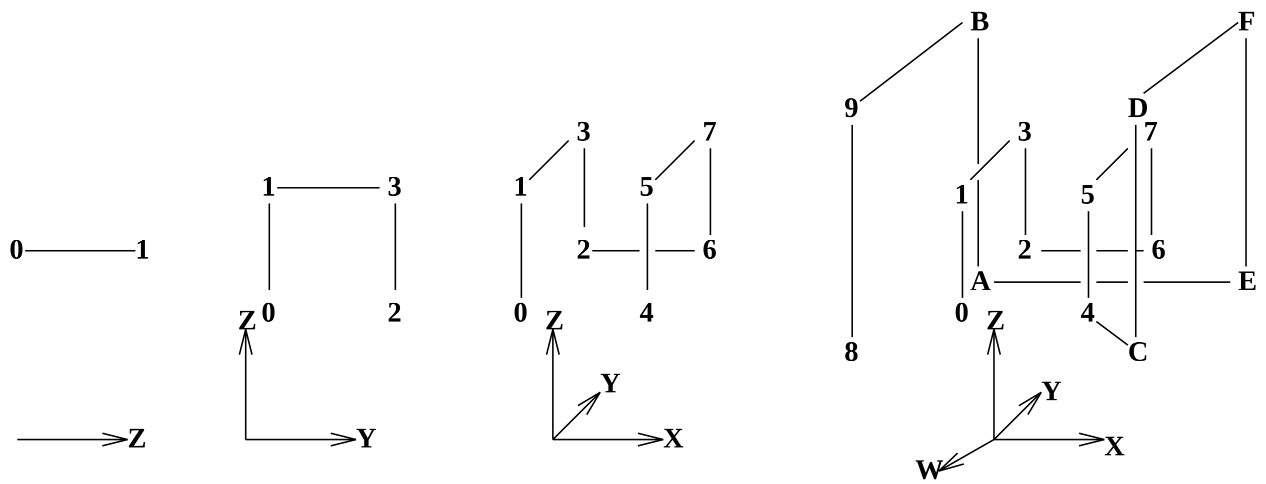

Figure 1: Bases for the Hilbert curves in 1 to 4 dimensions

$d = 1$	$d = 2$	$d = 3$	$d = 4$
0	00	000	0000
1	01	001	0001
	11	011	0011
	10	010	0010
		110	0110
		111	0111
		101	0101
		100	0100
			1100
			1101
			1111
			1110
			1010
			1011
			1001
			1000

(The effect also is that the rightmost bit is changed that yields a new code.) Figure 1 shows the connected hypercubes in one to four dimensions, using right-handed axes, with the axes corresponding to the bits in order $[[[w]x]y]z$. The vertices in the figure are labelled not with their Hilbert order but in the order induced by interpreting the bit codes as integers. (This is the basis for another space-filling curve, the Peano curve[4], or, in more than two dimensions, Z-order[3], so named to reflect the shape of the base curve.)

Note that the base curve starts and ends on nearest neighbours, namely points `00..0` and `10..0`, i.e., 0 and 2^{d-1} in the enumeration.

We will need to be able to calculate the Gray code, g, given its position, s, in the sequence, and vice-versa. Any Gray code in the latter half of the sequence (i.e., s has its first bit on) will have the first bit of g on. If this is the case, the complements of the remaining bits give the reflected position.

Algorithm 1 *Mapping from sequence number, s, to Gray code, g.*

1. Set the most significant bit of $g =$ the most significant bit of s.

2. If this is 0, recurse on the remaining bits.

3. If it is 1, toggle the remaining bits, then recurse. □

(To "toggle" a bit means to negate its value, i.e., `0` $\leftrightarrow$ `1`.)

The inverse mapping is only a litle more complicated. (It does not yield a Gray code.) Instead of toggling the remaining bits after a `1` is encountered, we alternately look for a `1` then a `0` then a `1`, and so on, to decide whether to toggle the remaining bits.

Algorithm 2 *Mapping from Gray code, g to sequence number, s.*

1. Set *control value* to `1`.

2. Set the most significant bit, b, of $s =$ the most significant bit of g.

3. If *control value* $=$ `1`
   - if b is 0, recurse on the bits following b;
   - if b is 1, toggle *control value*, toggle the bits following b, then recurse;

4. else (*control value* $=$ `0`)
   - if b is 1, recurse on the bits following b;
   - if b is 0, toggle *control value*, toggle the bits following b, then recurse. □

Naive implementation of both of these algorithms has complexity $O(d^2)$, but a linear implementation is not difficult.

The Gray code is the Hilbert curve on the basic hypercube, and the mapping to and from the sequence number is the mapping between Hilbert order and Z-order on this hypercube. Coordinates in d dimensions can be converted to Z-order by interleaving the bits, and vice-versa.

Now that we have this base step, we can go on to the recursion step, which replaces each vertex in the hypercube by a Hilbert curve on a hypercube at finer resolution.

2 Recursion

The recursion step is to replace each vertex in the base by a reflected copy of the base. This step will be repeated with the new vertices thus generated until the desired resolution is obtained.

We must determine which reflections are needed for each vertex. Here are the new curves to be generated for the 1- to 4-dimensional examples, using the enumeration as a shorthand for the bit codes.

$d=1$	$d=2$	$d=3$	$d=4$
$0 \to 01$	$0 \to 0231$	$0 \to 04623751$	$0 \to$ 08A26EC45DF73B91
$1 \to 01$	$1 \to 0132$	$1 \to 01546732$	$1 \to$ 0198CD5467FEAB32
	$3 \to 0132$	$3 \to 01546732$	$3 \to$ 0198CD5467FEAB32
	$2 \to 3102$	$2 \to 31026457$	$2 \to$ 3102A89BFDCE6457
		$6 \to 31026457$	$6 \to$ 3102A89BFDCE6457
		$7 \to 67320154$	$7 \to$ 6732ABFECD980154
		$5 \to 67320154$	$5 \to$ 6732ABFECD980154
		$4 \to 51376204$	$4 \to$ 51376204C8AEFB9D
			C $\to$ 51376204C8AEFB9D
			D $\to$ CD9801546732ABFE
			F $\to$ CD9801546732ABFE
			E $\to$ FDCE64573102A89B
			A $\to$ FDCE64573102A89B
			B $\to$ AB3267FECD540198
			9 $\to$ AB3267FECD540198
			8 $\to$ 913BF75DC46EA208

Of course, there are patterns to these solutions, and they generalize to any number of dimensions. In the two-dimensional case we see that vertex 0 is replaced by the reflection of the base in the YZ diagonal, which means that bits y and z are swapped. Vertices 1 and 3 are replaced by the untransformed base. Vertex 2 is replaced by the Y reflection of vertex 0, which means a reflection of the base in the $-YZ$ diagonal, namely a toggling of bits y and z if and only if they are equal.

In three dimensions, we see corresponding reflections: XZ for vertex 0, XY for vertices 1 and 3, $-YZ$ for vertices 2 and 6, $-XY$ for vertices 7 and 5, and $-XZ$ for vertex 4.

These reflections can be derived from the endpoints desired for each reflected copy of the base. The endpoints of the base are 0 and 2^{d-1}, and vertex 0 must be expanded by a copy of the base with endpoints 0 and 1. In this reflection, 0 is fixed and 4 becomes 1. Swapping x and z bits does the job. We do not care what happens to the rest of the vertices and edges, as long as the transformation is still a Gray code. In two and three dimensions we can see that this is so, and all edges still connect nearest neighbours, because the transformation is a reflection. In general we can prove that swapping two bits preserves the Gray code property of the sequence.

Lemma 1 *A Gray code remains a Gray code after two bits are swapped.*

Proof We must show (1) that successive codes remain nearest neighbours, and (2) that no duplicates result, i.e., the mapping is one-to-one. Consider one term, t, and its predecessor, s. (In a binary reflected Gray code, which we start with, the last term is a nearest neighbout of the first and may be considered its predecessor.)

1. There are two cases. Either the two bits swapped have the same values in both t and s, or they do not. In the first case, swapping them makes no difference. In the second case, only one of the four bits can be different from the other three, by the nearest-neighbour property, and swapping the bits preserves this property.

2. Since we start with a set of codes without duplicates, swapping only exchanges two columns of bits, and we end up with no duplicate codes. □

We need the same result for reflections in the opposite diagonals and in any of the axes.

Lemma 2 *A Gray code remains a Gray code after two bits are toggled when they have equal values.*

Proof Similar. □

Lemma 3 *A Gray code remains a Gray code after one bit is toggled.*

Proof Similar. □

We call the reflections of these three lemmas "diagonal reflection", "back-diagonal reflection", and "axis reflection", respectively.

Since the two-dimensional base has endpoints 02, reflecting this to 01 is an yz swap, and so 0132 maps to 0231. Reflecting it to 32 toggles the yz bits iff $y = z$, so 0132 maps to 3012. The two vertices in between expand to the base itself.

In three dimensions, the endpoints must be 01, 02, 02, 37, 37, 64, 54 for vertices 0, 1, 3, 2, 6, 7, 5, 4, respectively. The transformations needed are swap xz, swap xy, toggle yz iff $y = z$, toggle xy iff $x = y$, and toggle xz iff $x = z$, for vertices 0, 1 and 3, 2 and 6, 7 and 5, and 4, respectively. These yield the reflections given above, and the Gray codes given at the beginning of this section.

For d dimensions, we can show that any transformation from endpoints $0e$ (where e is a nearest neighbour of 0, and hence a power of 2 exclusively between 0 and 2^d) to any other pair of nearest neighbours can be accomplished by some combination of the reflections in Lemmas 1–3.

Lemma 4 *The vertices 0 and $2^p, 0 < p < d$ can be mapped into any other vertex, v, and a nearest neighbour by some combination of the operations of Lemmas 1–3.*

Proof We map 0 to v by back-diagonal reflections, and one axis reflection if necessary, then reflect the image of 2^p about the image of 0 and the image of $2^d - 1$ so that it coincides with the new nearest neighbour.

The constructive proof is the following algorithm. □

Algorithm 3 *Reflect the d-dimensional Hilbert curve with endpoints 0 and 2^p to one with endpoints v and a given nearest neighbour.*

1. If the number of on-bits in the representation for v is even, toggle these bits in pairs. This is the mapping by back-reflections from 0 to v.

2. If the number of on-bits in the representation for v is odd, follow the above mapping by one axis reflection to toggle the leftover bit.

3. Suppose the image of 2^p in the above mapping differs from v at bit p'. Suppose the target for 2^p differs from v at bit q. (Both the image and the target are nearest neighbours of v and so each differs from it at only one bit position.) If $p' = q$, do nothing, else reflect p' into q without moving v:

   - if the bits at p' and q have the same value for v, swap them (diagonal reflection);
   - if the bits at p' and q have opposite values for v, toggle them wherever else they are equal (back-diagonal reflection). □

Now that we know how to reflect the basic hypercube so its Hilbert curve has any needed endpoints, we must consider the relationships between hypercubes when they recursively replace the vertices of the hypercube at higher resolution. This relationship between two nearest-neighbour hypercubes has three components: orientation, direction and position.

The orientation of the hypercube when it replaces a vertex is just the direction from start to end point of its Hilbert curve, which in turn determines the above reflections to achieve these endpoints.

The direction from the new start point to the new endpoint of the reflected base is determined by the direction from the vertex it is replacing to the next vertex in the parent.

The position of the new start point is determined by alignment. Thus, in two dimensions, the endpoints must be 01, 02, 02, and 32, respectively for vertices 0, 1, 3, 2. Going in the Z direction from vertex 0 to vertex 1 requires the expanded vertex 0 to end in the Z direction from its start point, 0. Alignment requires that the expanded vertex 1 starts at its position 0 (and not, say, 4, 2, or 6). Going in the Y direction from 1 to 3 requires that the expanded vertex 1 end in the Y direction from this start point.

Now we elaborate.

We use the *orientation* to determine the end point of a Hilbert curve from its start point. The direction from the start to the end point must be the same as either the direction to the successor hypercube or the direction from the predecessor hypercube. One of these two may be impossible to choose because of the position of the start point. (Clearly we must start the Hilbert curve with the latter choice and end it with the former.) The bit in the start point corresponding to the chosen direction is changed to give the end point: if the direction is positive, the bit must change from 0 to 1; if it is negative, the bit must change from 1 to 0; if either of these is impossible, the other direction must be used.

We use *direction* and *position* to determine the start point of the neighbour Hilbert curve from the end point of its predecessor. Direction fixes one bit of the result, by toggling the bit corresponding to the direction. Position determines the remaining bits, by copying them from old end point to new start point.

Here is the 3-dimensional example, starting the curve at 0 (bit `000`). The top-level sequence is 01326754, and we expand each of these vertices into start-end pairs as follows.

<table>
<tr><th>Top Vertex</th><th>Bits</th><th>Succ/Pred</th><th>Dir.</th><th>Start–End</th></tr>
<tr><td>0</td><td>000</td><td></td><td></td><td></td></tr>
<tr><td></td><td>|</td><td>s: Z+</td><td></td><td></td></tr>
<tr><td></td><td>001</td><td></td><td></td><td>01</td></tr>
<tr><td></td><td>|</td><td></td><td>d: Z+</td><td></td></tr>
<tr><td>1</td><td>000</td><td></td><td></td><td></td></tr>
<tr><td></td><td>|</td><td>s: Y+</td><td></td><td></td></tr>
<tr><td></td><td>010</td><td></td><td></td><td>02</td></tr>
<tr><td></td><td>|</td><td></td><td>d: Y+</td><td></td></tr>
<tr><td>3</td><td>000</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>s: Z-</td><td></td><td></td></tr>
<tr><td></td><td>|</td><td>p: Y+</td><td></td><td></td></tr>
<tr><td></td><td>010</td><td></td><td></td><td>02</td></tr>
<tr><td></td><td>|</td><td></td><td>d: Z-</td><td></td></tr>
<tr><td>2</td><td>011</td><td></td><td></td><td></td></tr>
<tr><td></td><td>|</td><td>s: X+</td><td></td><td></td></tr>
<tr><td></td><td>111</td><td></td><td></td><td>37</td></tr>
<tr><td></td><td>|</td><td></td><td>d: X+</td><td></td></tr>
<tr><td>6</td><td>011</td><td></td><td></td><td></td></tr>
<tr><td></td><td>|</td><td>p: X+</td><td></td><td></td></tr>
<tr><td></td><td>111</td><td></td><td></td><td>37</td></tr>
<tr><td></td><td>|</td><td></td><td>d: Z+</td><td></td></tr>
<tr><td>7</td><td>110</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>p: Z+</td><td></td><td></td></tr>
<tr><td></td><td>|</td><td>s: Y-</td><td></td><td></td></tr>
<tr><td></td><td>100</td><td></td><td></td><td>64</td></tr>
<tr><td></td><td>|</td><td></td><td>d: Y-</td><td></td></tr>
<tr><td>5</td><td>110</td><td></td><td></td><td></td></tr>
<tr><td></td><td>|</td><td>p: Y-</td><td></td><td></td></tr>
<tr><td></td><td>100</td><td></td><td></td><td>64</td></tr>
<tr><td></td><td>|</td><td></td><td>d: Z-</td><td></td></tr>
<tr><td>4</td><td>101</td><td></td><td></td><td></td></tr>
<tr><td></td><td>|</td><td>p: Z-</td><td></td><td></td></tr>
<tr><td></td><td>100</td><td></td><td></td><td>54</td></tr>
</table>

This calculation gives us the endpoints from which we can determine the reflection operations that in turn give us the expansion of each vertex into the sequences shown at the beginning of this section.

3 Mapping Between Coordinates and Hilbert Order

We can now put everything together to find the mapping between d-dimensional coordinates and Hilbert order. To go from coordinates to Hilbert order, we extract the Gray codes for hypercubes at ever finer resolution by dividing the coordinates by decreasing powers of 2, keeping the remainder for the next level of resolution. We convert the highest Gray code to its sequence number using Algorithm 2, and we reflect subsequent Gray codes by the appropriate operators, then convert the resulting Gray code to its sequence number. These sequence numbers are multiplied by decreasing powers of 2^d and added.

Here is the process in three dimensions, starting with coordinates (2,0,2).

i	2^i	Gray	8^i	refl	seq	8^i×seq
1	2	(1,0,1)	8		6	48
0	1	(0,0,0)	1	(1,1,0)	4	4

So the position of (2,0,2) ($= 2 \times (1,0,1) + 1 \times (0,0,0)$) in Hilbert order is 48+4=52. This can be checked against the 3-D Hilbert curve in Figure 2. (This figure shows second- and third-order two-dimensional Hilbert curves, and the second-order three-dimensional curve, with the distance between the first-order vertices exaggerated for clarity. In this figure, the vertex numbers correspond to the Hilbert order, and not, as before, to Z-order.)

We have used the fact that vertex (1,0,1) (6th in the base Gray code) in the lower-resolution Hilbert order is expanded by the $-XY$ reflection, which toggles bits x and y if they are the same. This converts (0,0,0) to (1,1,0), which is vertex 4 in the base Gray code.

In two dimensions, let's look at higher powers, for example, (6,2) in third order.

i	2^i	Gray	4^i	refl	seq	4^i×seq
2	4	(1,0)	16		3	48
1	2	(1,1)	4	(0,0)	0	0
0	1	(0,0)	1	(1,1)	2	2

So the position of (6,2) ($= 4 \times (1,0) + 2 \times (1,1) + 1 \times (0,0)$) in Hilbert order is 48+0+2=50. This can be checked against the third-order 2-D Hilbert curve.

In this higher-order example, there are two reflections. Vertex (1,0) (3rd in the base Gray code) expands with a $-YZ$ reflection, mapping (1,1) to (0,0). This vertex (0,0) expands with a further YZ reflection: the $-YZ$ reflection maps (0,0) to (1,1), and the YZ reflection makes no further change.

The process inverts, and we can follow both examples in reverse to go, respectively, from 50 to (6,2) and from 51 to (2,0,2).

Algorithm 4 *Mapping from d-dimensional coordinates to Hilbert order*

coord2Hilbert(*dim, order, coord, xform, sum*)
(*xform* contains reflections, is initially the identity; *coord* is coordinates for one *dim*-dimensional point)

1. *gray* <– *coord*/2^{order}
2. *image* <– apply *xform* to *gray*
3. *seq* <– sequence from Gray (Algorithm 2)
4. *sum* <– *sum* + *seq*×$2^{dim \times order}$
5. return if *order*=0 then *sum* else
   coord2Hilbert(*dim, order*–1, *coord* **mod** 2^{order},
   (*xform* followed by the reflections corresponding to *seq*), *sum*)

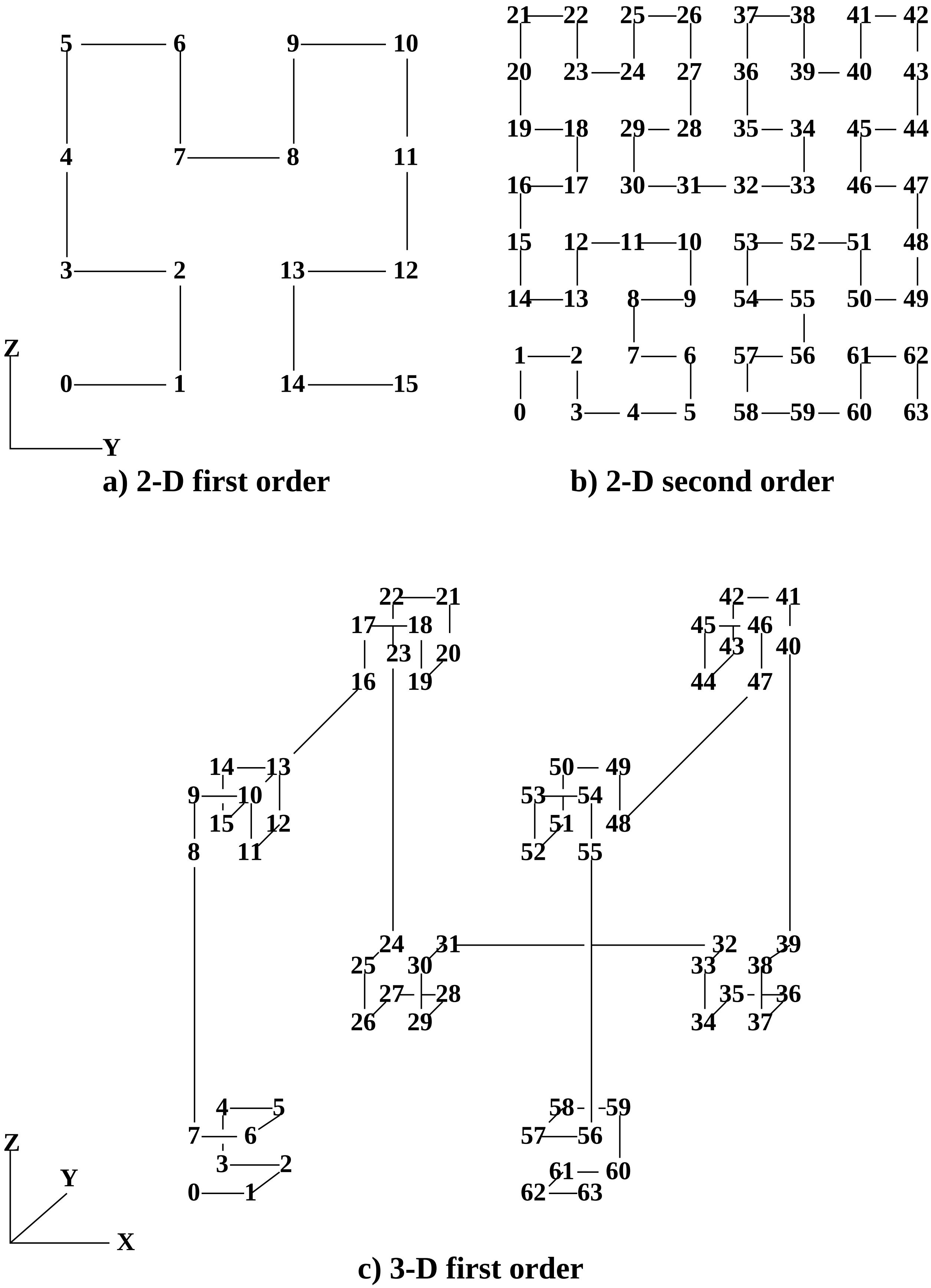

Figure 2: Example Hilbert Curves in 2 and 3 Dimensions

Algorithm 5 *Mapping from d-dimensional Hilbert order to coordinates*

Hilbert2Coord(*dim, order, pos, xform, cum*)
(*xform* contains reflections, is initially the identity; *pos* is position of one *dim*-dimensional point)

1. $seq <- pos/2^{dim \times order}$

2. $gray$ <– Gray from sequence (Algorithm 1)

3. $image$ <– apply $xform$ to $gray$

4. $cum <- cum + image \times 2^{order}$

5. return if $order$=0 then cum else
   Hilbert2Coord(dim, $order-1$, pos **mod** $2^{dim \times order}$,
   ($xform$ followed by the reflections corresponding to seq), cum)

References

[1] F. Gray. Pulse code communications. U. S. Patent 2 632 058, March 1953.

[2] D. Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Math. Ann.*, 38:459–60, 1891.

[3] J. A. Orenstein and T. H. Merrett. A class of data structures for associative searching. In *Proc. Third ACM SIGACT–SIGMOD Symposium on Principles of Database Systems*, pages 181–90, April 1984.

[4] G. Peano. Sur une courbe, qui remplit toute une aire plane. *Math. Ann.*, 36:157–60, 1890.