A Generalized 2D and 3D Hilbert Curve

Jakub Červený, Zzyv Zzyzek

Abstract

The two and three dimensional Hilbert curves are fractal space filling curves that map the unit interval to the unit square or unit cube. Hilbert curves preserve locality, keeping distance from the unit interval to their respective higher dimensional embeddings. Finite approximations of the Hilbert curve can be constructed in stages by stopping the recursive construction at a specified depth, providing locality from a discrete array of points to points in the embedded dimension. The locality property of the Hilbert curve can help with caching optimizations based on order and finds uses as a companion method in applications ranging from job scheduling to scene rendering. One limitation of the Hilbert curve approximation construction is that the regions need to be exact powers of two, making it difficult to work with in many real world scenarios. In this paper, we present the Gilbert curve, a conceptually straight forward generalization of the Hilbert curve, that works on arbitrary rectangular regions. The construction provides reasonable worst case run-time guarantees for random access lookups for both two and three dimensions. We also provide experiments showing comparable quality in locality of the Gilbert curve to the Hilbert curve for a range of cuboid regions.

1. Introduction

1.1. Overview

We present the *Gilbert curve*, a generalized Hilbert curve for 2D and 3D, that works on arbitrary rectangular regions.

An overview of the benefits of the Gilbert curve are that it is:

- Valid on arbitrary rectangular regions
- Efficient at random access lookups ($O(\lg N)$)
- A conceptually straight forward algorithm
- Able to realize curves without notches unnecessarily and limits the resulting curve to a single notch when forced

Further, we show:

- Equivalent to the Hilbert curve when dimensions are exact powers of 2
- Measures of locality are preserved (Section X)
- Comparison metrics to other space filling curves (Section X)

Some drawbacks are that:

- Our implementation is recursive, which may be undesirable for some applications requiring better than $O(\lg N)$ memory usage or better constant bounds for runtime performance
- Might not adequately capture some features of a Hilbert curve

Generalizations of the Hilbert curve to non power of two rectangular cuboid regions have been explored before but are overly complicated algorithmically, create unbalanced curves and often don't generalize well to 3D. Our realization focuses on a conceptually simple algorithm which creates pleasing resulting curves and works in both 2D and 3D.

Space filling curves are a specialization of a more general Hamiltonian path, but have a more stringent requirement of local connectivity. The local connectivity, or locality, preserves a measure of nearness, where points from the source unit line remain close in the embedded space.

The Gilbert curve algorithm works by choosing sub rectangular cuboid regions to recursively find a connecting path. If one extent of the cuboid becomes too large or small relative to the other lengths, a special subdivision is performed to try and make further subdivisions more cube-like. Subdivisions are performed until a base case is reached and the lowest unit of the curve can be realized.

A Hamiltonian path is not always realizable for certain side lengths and endpoint constraints. In such a case, the Gilbert curve will introduce a single diagonal path move, called a *notch*.

1.2. Definitions

A Gilbert curve is a self avoiding path, restricted to the cubic lattice graph in a rectangular region, that also has locality. That is, we want to find a Hamiltonian path, with pre-specified start and end points, that hit every vertex within a cuboid rectangular region exactly once with any points close on the curve being appropriately close in the cubic lattice.

Here, the *cubic lattice graph* is a graph whose vertices are the integral points in $\mathbb{Z}^3$, with edges joined between them when they are unit distance away.

A discrete *path* is an array of points, $\omega = (\omega_0, \omega_1, \ldots, \omega_{N-1})$, where each w_i is on the cubic lattice graph ($w_i \in \mathbb{Z}^3$). A *self avoiding path* ω, is one that does not visit a previously vertex ($i \neq j \rightarrow \omega_i \neq \omega_j$). A *Hamiltonian path* is a valid path through a graph that visits every vertex exactly once, starting and ending at pre-specified points.

A *cuboid* is a rectangular region of side lengths (N_x, N_y, N_z), of total vertices $N = (N_x \cdot N_y \cdot N_z)$. A cuboid region is said to have a *feasible* Hamiltonian path if there exists a Hamiltonian path, with endpoints specified, that hits every vertex within the cuboid region exactly once, starts at the specified start point, ω_0 and ends at the specified endpoint ω_{N-1}. A cuboid region that cannot have a Hamiltonian path, with the specified endpoints, is said to be *infeasible*.

A self avoiding path of length N that is confined to the rectangular region (N_x, N_y, N_z) is necessarily a Hamiltonian path.

A discrete realization of the Hilbert curve as a path within a cuboid region has an additional *local connectivity* feature. For our purposes, *local connectivity*, or *locality*, denotes the idea that if $\text{dist}_1(i, j)$ is small, then $\text{dist}_d(w_i, w_j)$ should also be small by some measure, where $\text{dist}_d(\cdot, \cdot)$ is a distance function defined for dimension d. The condition of

https://github.com/jakubcerveny (J. Červený)
0009-0005-2175-1166 (Z. Zzyzek)

a Hamiltonian path through a cuboid region doesn't address locality and the Gilbert curve has additional features to make its locality similar to the Hilbert curve. We use metrics to get a handle on local connectivity in a discrete and finite setting and leave their definition and usage until Section 5.

The concept of local connectivity gives one of the key characteristics of a space filling curve and differentiates it from other curves that can fill rectangular regions but don't do so in a locally connected way [1]. The local connectivity, or locality, is the feature that allows clustering in dimension one to remain clustered in higher dimensions.

If a cuboid region has one side larger or smaller than the maximum or minimum of the remaining two, respectively, for a given ratio, ρ, it is said to be *eccentric*. For example is, if $\rho = \frac{3}{2}$ and $N_x > \frac{3}{2}\max(N_y, N_z)$, the cuboid region is said to be eccentric.

In this paper, we talk about *shape templates* as cuboid subdivision schemes, often with some choice as to whether to make the side lengths of the subdivided regions even or odd. For 2D, we use the term U-split to talk about the shape template used. For 3D, we use the term J-split to talk about the bulk recursion shape template and an S-split to talk about the shape templates used for eccentric cuboids.

2. Related Work

Peano discovered the first space filling curve, with Hilbert later discovering a space filling curve that we concern ourselves with in this paper [1]. Hilbert's curve takes a one dimensional line and maps it to a compact, connected and locally connected set. We only touch on the technical definitions in this paper but the reader is referred to Sagan [2] for a more thorough introduction. An introduction to the space filling curves with a focus on the Hahn-Mazurkiewicz theorem, the sufficient and necessary conditions to be a space filling curve, can be found in Kupers [3].

The locally connected property of space filling curves provide a condition where nearby index points in the domain lead to scaled nearby points in the output embedded dimension. The *locality* feature means that nearby index points cluster in the mapped higher dimensional space. This feature can be used to highlight one dimensional clustered data in higher dimensions.

Wiener noticed that Lebesgue integration could be reduced from higher dimensional integration to one-dimensional integration by the use of space filling curves [4]. Lera and Sergeyev have investigated reducing higher dimensional global optimization on functions satisfying the Lipschitz condition, reducing the multi dimensional optimization to a single dimension through a space filling curve mapping [5]. Asano et al. discuss space filling curves and their uses in geometric data structures to create localized spatial access patterns [6], Böhm, Perdacher and Plant show speedups in Cholesky decomposition and the Floyd-Warshall algorithm by reducing cache misses from a space filling curve data access schedule [7]. Niedermeier discusses a new method of H-indexing as a benefit over a Hilbert indexing scheme to create optimal locality for mesh indexing in [8].

[1] For example, the *zig-zag* curve that creates a straight line until it hits the bounds of the region to turn around and increment in the other dimensional lengths to fill a rectangular region, is not space filling as it isn't locally connected.

Munroe used the locality property to create a stylized mapping of the internet protocol (IPv4) space [9]. Anders used a Hilbert curve mapping to visualize the human genome as a 2d map [10]. Cortesi has used a Hilbert curve to visualize binary files [11]. Dafner, Cohen-Or and Matias use a space filling curve that is adapted to the image context to improve autocorrelation above what can be achieved with a Peano or Hilbert curve [12].

Hamilton and Rau-Chaplin explore creating Hilbert curves with unequal side lengths in [13] but require the side lengths to still be exact powers of two. Rong, Zhang and Lin [14] proposed a generalization of the 2D Hilbert curve to arbitrary rectangular regions that involves subdividing smaller regions into nearest powers of to areas until a base case is reached. They extend their method to 3D but require the depth dimension to be an exact power of two.

In a website application and scanned note [15], Tautenhahn provided the basis for a generalized 2D space filling curve. Tautenhahn also included policies for when to subdivide regions preferentially in only one dimension, what we call *eccentric splits*. Tautenhahn's exploration details the parity arguments necessary for when a subdivision scheme can be employed without creating diagonal moves (*notches*). In this paper, we extend Tautenhahn's ideas to create a 2D and 3D generalized Hilbert curve, employing parity arguments and conditions under which eccentric splits should be employed.

For recursive space filling curves, there is a choice in recursive or subdivision schemes. Haverkort catalogs different techniques for constructing 3D Hilbert curves in [16] and [17].

Though all space filling curves must fill the domain space and maintain locality, features or fine grained qualities might differ between subdivision schemes. Mokbel et al. provide some performance statistics of different recurrence schemes in [18]. Gotsman and Lindenbaum provide more detailed definitions of locality, explore their definition of locality for different space filling curves and conclude that the Hilbert curve is close to optimal [19]. Haverkort and Walderveen provide a bounding-box quality metric for 2D space filling curves in [20]. In [21], Haverkort provides a concept of an Arrwwid number that measures how best a volume can be covered by curve sections and discusses optimal choices of space filling curves based on this metric. Voorhies defines the concept of *coherence* to measure how many times a ray passes through a volume to differentiate between zig-zag curves, Peano curves and Hilbert curves [22].

Though recursive subdivision methods can be made to run in $O(\log N)$ time and space ($N \sim W \times H \times D$), this is undesirable under certain high performance conditions where a memory usage needs to be constant or where constant bounds matter. Butz described a non-recursive, byte-oriented algorithm for Hilbert's space filling curve in [23]. Moore later extended Butz's idea to create a fast non-recursive algorithm that includes range queries [24]. Jia et al. provide further enhancements for efficient the 3D Hilbert curve encoding and decoding in [25]. In Holzmüller's bachelor thesis, an algorithm for average case $O(1)$ neighbor queries is shown [26].

For a more general treatment on self avoiding walks, the reader is referred to Madras and Slate [27]. Though not directly connected to space filling curves, the reader is referred to Christensen and Moloney [28] for methods used in data analysis and visualization used in this paper.

3. Algorithm

3.1. Overview

We discuss one possible generalization of the Hilbert curve, which we call the Gilbert curve, to arbitrary axis-aligned cuboid regions.

The design of the Gilbert curve is done to provide a:

- Conceptually simple algorithm
- Harmonious realization
- Resulting curve with good locality conditions

The discretized Hilbert curve recursively traces out a Hamiltonian path through a square region whose sides are powers of two. The Gilbert curve extends this idea to trace out Hamiltonian paths through to arbitrary axis-aligned cuboid regions.

The Gilbert curve recursively subdivides a larger cuboid region into smaller cuboid regions of different sizes and orientations. The subdivision scheme uses a shape template that will be explained in more detail in subsequent sections. The orientations of the subdivided cuboid regions are chosen so that the recursive path will start and end at pre-specified endpoints, connecting neighboring cuboid regions after a path is realized.

If a cuboid becomes too oblong, or *eccentric*, a subdivision shape template scheme is chosen in an attempt to create subdivisions that are closer to being cube like.

The subdivided cuboid region is processed by the Gilbert curve algorithm, with virtual origin point $p \in \mathbb{Z}^3$ and local coordinate system $[\alpha, \beta, \gamma]$, with $\alpha, \beta, \gamma \in \mathbb{Z}^3$. Each of α, β and γ are axis aligned and orthogonal.

In a cuboid region, the Hamiltonian path starts at p and ends at $p + \alpha$. For this reason, we call α the "width-like" dimension, with β and γ called the "height-like" dimension and "depth-like" dimension, respectively.

When processing the subdivided cuboid regions, the coordinate system is updated with new integral lengths and rotated. Since rotations are only in units of $(\pi/2)$ radians and α, β, γ start off as axis-aligned and orthogonal, they remain axis aligned and orthogonal throughout.

When applying the shape template to choose the subdivided cuboid regions during the course of the algorithm, side lengths are chosen so as to preserve the possibility of a Hamiltonian path. Should the lengths of the cuboid being subdivided preclude a Hamiltonian path from occurring, the lengths of the subdivided cuboids are chosen so that all but one will preserve the possibility of a Hamiltonian path. In this case, since only a single cuboid absorbs the impossibility of a Hamiltonian path, and this process is recursively applied, there will be a single jump, or *notch*, in the resulting curve, globally.

The next sub-section discusses parity arguments for when a valid curve can be traced out in a sub-region. We then discuss in detail the 2D Gilbert curve algorithm, move on to a discussion of the 3D Gilbert curve and end with a general discussion on extensions to the algorithm, special cases and other pathological cases.

When talking about the 2D Gilbert curves, we assume vectors are in $\mathbb{Z}^3$ so they can be used without alteration for algorithms working in 3D.

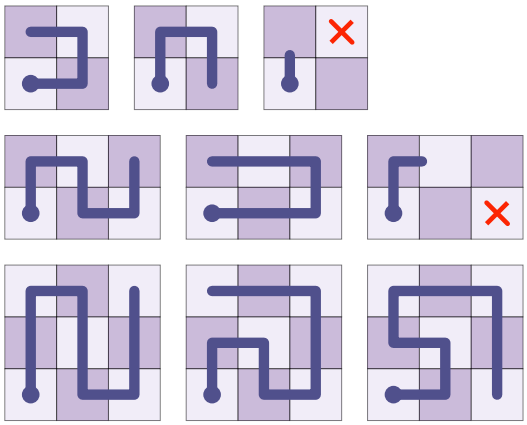

Figure 1: Illustrative examples of Hamiltonian paths height/width that are even/even, even/odd and odd/odd, respectively, when starting from the lower left hand corner

3.2. Valid Paths from Grid Parity

The feasibility of determining whether there exists a Hamiltonian path in a rectangular cuboid grid region can be accomplished through parity arguments. Label grid cell points in a volume as 0 or 1, alternating between labels with every axis-aligned single step move. Any Hamiltonian path that ends at one of the three remaining corners has to have the same parity as the starting point if the volume is odd, or different parity if the volume is even.

Path Possible		Volume	
		even	*odd*
$\lvert\alpha\rvert$	*even*	Yes	Yes
	odd	**No**	Yes

Table 1
Table showing when a Hamiltonian path is possible. Here, $|\alpha|$ is the absolute difference in start and end position of the path which coincides with one of the axis aligned side length of the cuboid volume. A Hamiltonian path is possible only when $|\alpha|$ is even or both $|\alpha|$ and the volume are odd.

For a path starting at $(0, 0, 0)$ and ending $|\alpha|$ steps in one of the axis-aligned dimensions, then Table 1 enumerates this condition under which a valid path is possible. Figure 1 illustrates this for starting position $(0, 0)$ with volumes (2×2), (3×2) and (3×3), where a red cross indicating a precluded endpoint.

Without loss of generality, we will assume a curve starts from position $p_s = (0, 0, 0)$ and has proposed endpoint at $p_e = ((w - 1), 0, 0)$, with a cuboid region as $\alpha = (w, 0, 0), \beta = (0, h, 0), \gamma = (0, 0, d)$. We state, without proof, that a Hamiltonian path is always possible from p_s to p_e when $|\alpha|$ is even or when all of $|\alpha|$, $|\beta|$ and $|\gamma|$ are odd $(|\alpha| \cdot (1 - |\beta| \cdot |\gamma|) \equiv 0 \bmod 2)$.

With this condition, any cuboid subdivision will always have a Hamiltonian path within it and we can recreate a Hamiltonian path in the larger cuboid by connecting endpoints from the ending point of one cuboid to the starting point of the succeeding cuboid. For cuboids that violate this condition, there will be a required notch and an feasible cuboid subdivision is possible for all but one of the cuboid subdivisions, recursively limiting the notch violation to a single point.

3.3. 2D Generalized Hilbert Function (Gilbert2D)

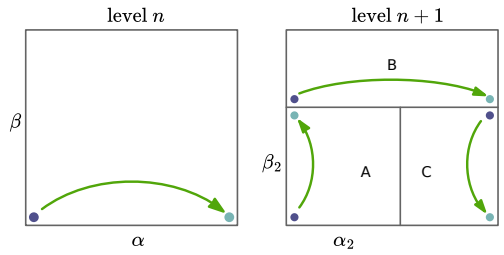

Figure 2: For the 2D Gilbert curve, the main rectangle is subdivided into three sub regions, making sure their paths connect and preferring even side lengths for the first rectangular subdivision. Partitioning the rectangle in this way is what we call a U-split.

For the 2D Gilbert curve, a U-split template is used, highlighted in Figure 2. The U-split breaks the region into three sub-blocks, labeled A, B and C. Each of the A, B and C regions have local coordinate systems that are resized, rotated and new endpoints chosen so the process can be recursively re-applied.

The sides of the subdivided blocks A, B and C are chosen to remain integral. The width-like dimension for subdivided block A is chosen to be even (β_2 in Figure 2) by dividing the height-like dimension of the original region (β in Figure 2) by two and by adding one if need be. Since the width-like length of the subdivided A and C block is even (β_2), there is a guaranteed valid Hamiltonian path in both.

Should the original width-like length be even (α in Figure 2), the C block will continue to have a valid Hamiltonian path. If the original width-like dimension is odd, a Hamiltonian path is not possible and there is a forced notch that will appear in the subdivided block C.

Figure 3 gives examples of the different parity conditions for the original area, as well as the different conditions when the A and C subdivided areas are coerced into having even parity. Figure 3 also shows when a Hamiltonian path is not possible, indicated by a red cross, and illustrates how the notch is pushed into the C block under these conditions.

If a subdivided block becomes too long in its width-like dimension, the block is divided in half and the recursion proceeds as normal. Even sides are only enforced if the length is larger than 2, with a length 2 or smaller representing the base case. The base case enumerates paths in the axis direction of length greater than one.

Since A and C are chosen to have an even width-like local axis length, no new notches are introduced and any obligatory notch is effectively "pushed" into the C block.

Algorithm 1 shows the pseudo-code for computing the 2D Gilbert curve. Note that α and β are taken to be vectors in 3D, where the third dimension can be ignored if a purely 2D curve is desired. The generalization to 3D allows the Gilbert2D function to be used unaltered when the 3D Gilbert curve needs to trace out in-plane sub-curves.

The $\delta(\cdot)$ function returns one of the six directional vectors indicating which of the major signed axis aligned directions the input vector points to $((\pm1,0,0),(0,\pm1,0),(0,0,\pm1))$. For completeness, the function is defined in Procedure 4 in Appendix B.1.

Algorithm 1 assumes standard Euclidean two norm ($|v| = \sqrt{v_0^2+v_1^2+v_2^2}$) and abuses notation by allowing scalar to vector multiplication ($i \in \mathbb{Z}, v \in \mathbb{Z}^3, i \cdot v \to (i \cdot v_0, i \cdot v_1, i \cdot v_2)$). where the context is clear.

Algorithm 1 2D Generalized Hilbert Function (Gilbert2D)

```
# p, α, β ∈ ℤ³
function GILBERT2D(p, α, β)

    α₂, β₂ = div(α, 2), div(β, 2)

    if (|β| ≡ 1) then
        yield p + i · δ(α) forall i ∈ |α|

    else if (|α| ≡ 1) then
        yield p + i · δ(α) forall i ∈ |β|

    else if (2|α| > 3|β|) then
        if (|α₂| > 2) and (|α₂| mod 2 ≡ 1) then
            α₂ ← α₂ + δ(α)
        end if

        yield Gilbert2D(p,
                        α₂, β)

        yield Gilbert2D(p + α₂,
                        α − α₂, β)

    else
        if (|β₂| > 2) and (|β₂| mod 2 ≡ 1) then
            β₂ ← β₂ + δ(β)
        end if

        yield Gilbert2D(p,
                        β₂, α₂)

        yield Gilbert2D(p + β₂,
                        α, (β − β₂))

        yield Gilbert2D(p + α − δ(α) + β₂ − δ(β),
                        β₂, −(α − α₂))
    end if
end function
```

Example outputs of the *Gilbert2D* function are listed in Figure 4, showing realizations for dimensions (8×8), (18×6), (13×8) and (14×14). The (8×8) Gilbert curve in Figure 4 (i) is identical to the Hilbert curve of side length 8. The (18×6) (ii) curve highlights a horizontal split when the α, or width-like, dimension becomes $\frac{3}{2}$ larger than the β, or height-like, dimension. For the (13×8) (iii) Gilbert curve, a Hamiltonian path is not possible and the algorithm inserts a notch in a best effort approach to realize the curve.

3.4. 3D Generalized Hilbert Function (Gilbert3D)

The concept of a U-split is extended to 3D by constructing J-split shape templates. An exploded view of the J_0 shape template is shown in Figure 5.

As discussed in section 3.1, we consider $p \in \mathbb{Z}^3$ to be the local origin point in the cuboid with $\alpha, \beta, \gamma \in \mathbb{Z}^3$ to be the local basis whose lengths encode the size of the cuboid. The generalized 3D Hilbert algorithm (*Gilbert3D*) is shown in Algorithm 2.

Side lengths of the subdivided cuboids are chosen to try

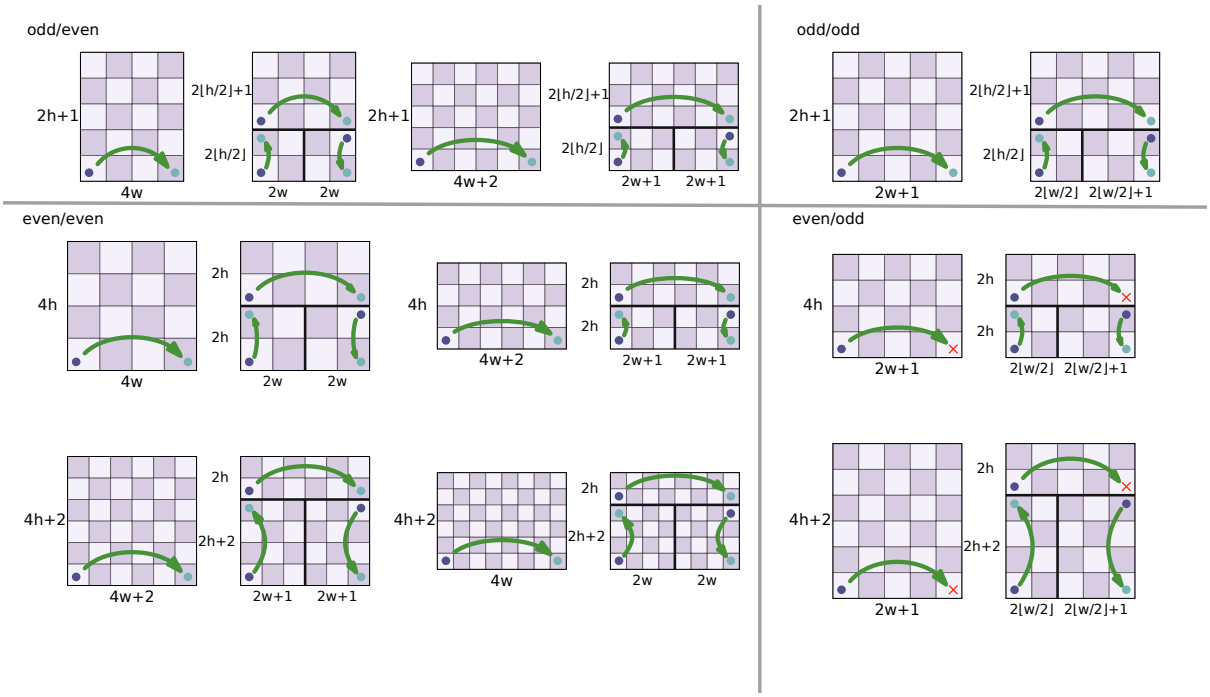

Figure 3: Enumeration of the subdivision template depending on different parities of α and β dimensions.

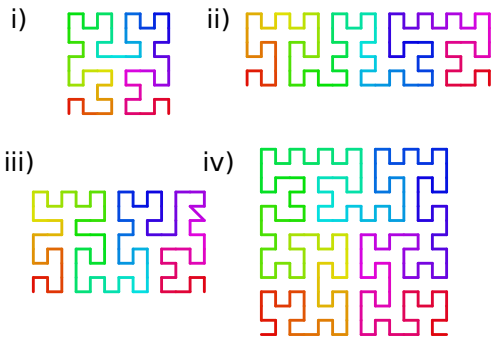

Figure 4: Example outputs of the 2D Gilbert curve algorithm for width and height of i) 8×8 ii) 18×6 iii) 13×8 and iv) 14×14. The 8×8 Gilbert curve (i) is identical to the Hilbert curve of length 8. For the 13x8 curve (iii), there is no Hamiltonian path possible from $(0, 0)$ to $(12, 0)$ resulting in the Gilbert curve algorithm inserting a notch in the upper right hand corner.

and not preclude a Hamiltonian path, if possible. If a path starts at the local $(0, 0, 0)$, a Hamiltonian path that ends at $((w-1), 0, 0)$ is only possible when w is even or when all lengths are odd, including w. That is, only when $(|\alpha| \cdot (1 - |\beta| \cdot |\gamma|) \equiv 0 \mod 2)$ will a Hamiltonian path be possible through the cuboid region.

Should a Hamiltonian path not be possible, all but one sub-cuboid regions are chosen to allow a Hamiltonian path. The sub-cuboid where a Hamiltonian path is not possible will be further subdivided as necessary, recursively ensuring that the infeasible Hamiltonian path region will eventually be resolved at a single point.

When considering a cuboid region to trace out a Gilbert curve on, there are conceptually three main guides:

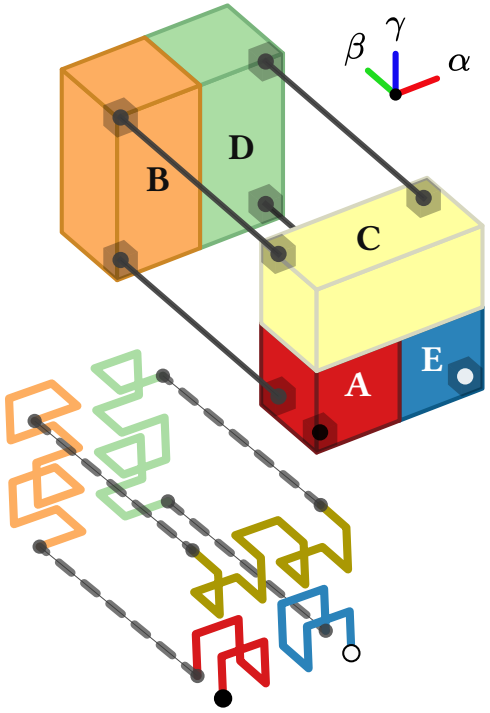

Figure 5: The J_0-split subdivision, representing the main subdivision of the bulk recursion for the 3D Gilbert curve case.

- *Bulk Recursion Case:*
  The cuboid region defined by α, β, γ is sufficiently cube-like
- *Eccentric Case:*
  One of $|\alpha|, |\beta|, |\gamma|$ are significantly larger or smaller than the other two dimension
- *Base Case:*

Algorithm 2 3D Generalized Hilbert Function (Gilbert3D)

```
# p, α, β, γ ∈ ℤ³
function GILBERT3D(p, α, β, γ)

    # Parity of dimensions
    α₀ ← (|α| mod 2)
    β₀ ← (|β| mod 2)
    γ₀ ← (|γ| mod 2)

    # Base cases
    if ((|α| ≡ 2) and (|β| ≡ 2) and (|γ| ≡ 2))
        return Hilbert3D(p,α,β,γ)
    return Gilbert2D(p,β,γ) if (|α| ≡ 1)
    return Gilbert2D(p,α,γ) if (|β| ≡ 1)
    return Gilbert2D(p,α,β) if (|γ| ≡ 1)

    # Eccentric cases
    if (3|α| > 5|β|) and (3|α| > 5|γ|))
        return S₀(p,α,β,γ)
    if (2|β| > 3|γ|) or (2|β| > 3|α|))
        return S₂(p,α,β,γ)
    if (2|γ| > 3|β|)
        return S₁(p,α,β,γ)

    # Bulk recursion
    return J₀(p,α,β,γ) if (γ₀ ≡ 0)
    return J₁(p,α,β,γ) if (α₀ ≡ 0)or(β₀ ≡ 0)
    return J₂(p,α,β,γ)
end function
```

One of $|\alpha|, |\beta|, |\gamma|$ are length 1 or all are length 2

For the *bulk recursion case*, there are three main shape templates, J_0, J_1 and J_2. An atlas of which J-split to choose, what parity each of the subdivided cuboid regions, and how to connect each of the subdivisions, is shown in Figure 6. Figure 6 also enumerates the local coordinate system for each of the sub-cuboid regions, displayed under each of the exploded sub-cuboid region volumes. When a Hamiltonian path is infeasible, a red cross is shown on the cuboid region where the notch will be directed to. Pseudo-code for each of the J-split functions is provided in Appendix B.3

For many of the subdivisions, there is some choice of parity for non-constrained sides. For example, if all sides start out even (parity 0), the J_0 template would be used, constraining $|\beta_2|$ to be even but otherwise allowing $|\gamma_2|$ and $|\alpha_2|$ to be chosen as even or odd parity.

For conceptual simplicity, $|\beta_2|$ and $|\gamma_2|$ are preferred even. Should $|\alpha|$ be even, $|\alpha_2|$ is preferred even. Otherwise, in the case $|\alpha|$ odd, $|\alpha_2|$ is preferred odd.

When $|\alpha|$, $|\beta|$ or $|\gamma|$ become smaller than 2, coercing the respective sub-cuboid side lengths to be odd or even is not done, as the sub-cuboid region can be safely reduced to its lower dimensional base case.

The *base case* reduces to *Gilbert2D* if at least one of the dimensions is 1. If all lengths are exactly two, a $(2 \times 2 \times 2)$ Hilbert curve is created (*Hilbert2x2x2* in Appendix B.1).

In the *eccentric case*, where a cuboid region is too lopsided or oblong, an S-split is used to try to shape sub-cuboid regions to be more cube-like. Figure 7 shows the three shape templates, S_0, S_1, S_2, along with the connecting points, for the different conditions when one axis length is significantly different from the others. Pseudo-code is provided for each of the S-split functions in Appendix B.2.

When $|\alpha| > \frac{5}{3}\max(|\beta|, |\gamma|)$, the S_0 template is used to partition the cuboid the α axis. In the other cases, when $|\gamma| > \frac{3}{2}\max(|\alpha|, |\beta|)$ or $|\beta| < \frac{2}{3}\min(|\alpha|, |\gamma|)$ the S_1 template is used and the S_2 template is used when $|\alpha| < \frac{2}{3}\min(|\beta|, |\gamma|)$, $|\gamma| < \frac{2}{3}\min(|\alpha|, |\beta|)$ or $|\beta| > \frac{3}{2}\max(|\alpha|, |\gamma|)$.

Figure 8 shows curated example curves for a choice of different side lengths. Example *(i)* shows a Gilbert curve of $(4 \times 4 \times 4)$, reducing to a Hilbert curve of the same dimension. Examples *(ii)*, *(iii)* and *(iv)* show Gilbert curves of $(4 \times 4 \times 5)$, $(5 \times 5 \times 5)$ and $(6 \times 6 \times 6)$ showing a J_1, J_2 and J_0 split, respectively. An S_0, S_2 and S_1 split are showing in the *(v)*, *(vi)* and *(vii)* Gilbert curves of lengths $(8 \times 4 \times 4)$, $(3 \times 5 \times 3)$ and $(3 \times 3 \times 5)$ respectively. A Gilbert curve with an unavoidable notch is shown in *(viii)*, where the side lengths $(5 \times 4 \times 4)$ preclude a Hamiltonian path from $(0, 0, 0)$ to $(4, 0, 0)$.

The constants used to determine eccentricity are chosen by considering the *average defect* of the subdivided cuboid volumes. We use the term *defect* to mean the ratio of volume of a cuboid region to the cube of its smallest length. That is:

$$\text{defect}(|\alpha|, |\beta|, |\gamma|) \stackrel{\text{def}}{=} \frac{|\alpha| \cdot |\beta| \cdot |\gamma|}{\min(|\alpha|, |\beta|, |\gamma|)}$$

For a cube, the defect will be 1, with an increasing defect the larger a side length becomes relative to the other two.

The *average defect* is defined to be the sum of the individual sub-cuboid defects, weighted by each sub-cuboids proportional volume.

The concept of defect is ad hoc and has potential problems[2]. Regardless, with this ad hoc definition, we can justify constants used for the eccentric cases by showing a defect reduction, relative to the bulk recursion case.

For the S_0 template, the two subdivided cuboids are chosen to be roughly half $|\alpha|$. For the other templates making the obvious choice of subdividing the "height-like" axis by half does not lead to a defect reduction. That is, choosing a roughly half subdivision of $|\gamma|$ for S_1 or a roughly half subdivision of $|\beta|$ for S_2, does not always yield a defect reduction.

A constant of $\frac{1}{3}$ is chosen to split the "height-like" dimensions for the S_0 and S_1 case, while the other split is chosen at the halfway mark. The calculations showing that the constants chosen yield a defect reduction in each of the three eccentric S-splits is provided in Appendix A.1.

3.5. Index to Point Mapping Functions (d2xyz, xyz2d)

To provide an index to point lookup or to convert a point to an index along the curve, Algorithm 1 and Algorithm 2 can be modified. Since the recursion is on cuboid regions, we can adjust an index or point easily by skipping over the appropriate number of cells.

A function to convert from an index along the Gilbert curve to a point in space is provided as *Gilbert2D_d2xyz* in Appendix 11. A destination and current index is added to the function call, where the destination index is tested against the current updated index to test whether to continue with the recursion. Otherwise, cuboid regions are

[2]For example, if one dimension is of length 1, the denominator will always be 1. Further, under this definition, the J-split will lead to an increase defect even though another recursion level for a J-split on the eccentric cuboids would lead to a near equal defect, indicating a recursive definition might be better.

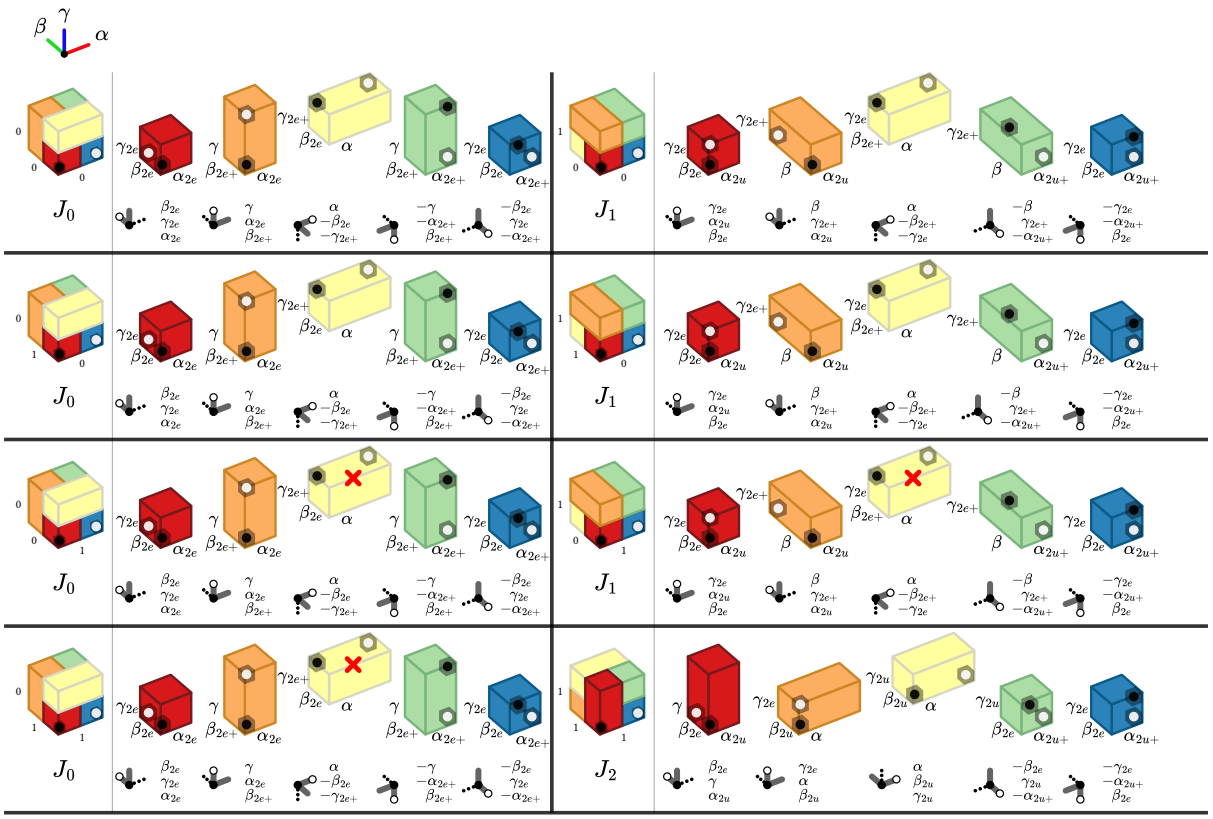

Figure 6: Bulk recursion J-split atlas for the 3D Gilbert algorithm. A subscript of $2e$ is used to denote a preference to coerce the length even where as the subscript $2u$ is used to denote a preference to coerce the length odd. The subscript $2e+$ is the remainder of the axis after removing the $2e$ portion (e.g. $\alpha_{2e+} = \alpha - \alpha_{2e}$). The subscript $2u+$ is the remainder of the axis after removing the $2u$ portion (e.g. $\beta_{2u+} = \beta - \beta_{2u}$). The parity of the original cuboid volume are shown in the left most, assembled, volume, for each of the rows. The local axis for each of the subdivided cuboid regions is shown underneath them, with a white dot denoting the "width-like" axis, a solid line denoting the "height-like" dimension and a dotted line denoting the "depth-like" dimension. For each of the cuboids, the block dot denotes the start of the path and the white dot denotes the path end. A red cross is used to show when a Hamiltonian path is not possible within a cuboid volume.

skipped without processing, providing an $O(\log N)$ worst case run time.

A similar function to convert from a point in space to an index along the curve is called *Gilbert2D_xyz2d* and is provided in Appendix 12. *Gilbert2D_xyz2d* functions similarly to *Gilbert2D_d2xyz* but instead uses a current test point, q, to test whether the recursion is within the cuboid bounds and, if so, recurs. As with the *Gilbert2D_d2xyz*, *Gilbert2D_xyz2d* provides a worst case execution of $O(\log N)$ for point to index lookups.

For the sake of brevity, the **xyz2d** and **d2xyz** functions are not provided for 3D but the reader is referred to the reference implementation for details [3].

3.6. Adaptive Extensions

Applications that require a strict Hamiltonian path without a notch point, but are unconstrained in endpoint location, can choose endpoints on the even length dimension so that a Gilbert curve with a Hamiltonian path realization is possible. Other options can include increasing one or more length dimensions with padding to make the major width-like axis (α) even or by forcing all length dimensions to be odd.

In some cases, choosing the width-like axis, α, on the lone basis of whether a Hamiltonian path is possible might lead to a Gilbert curve that isn't visually pleasing. For example, in the extreme case when $|\alpha| = 2$ and $|\beta|$ odd ($|\gamma| = 1$), a long "U" path will occur, with two long straight lines running in parallel. In this case, a notch might be a reasonable compromise to maintain a more pleasing fold density.

A more general adaptive extension can be done by choosing the base axis, α, to be the longest side length to try and make the initial subdivided cuboids as cube like as possible.

3.7. Corner Cases, Pathological Cases

For the special case when dimensions are of the form $((2k+1), 2)$, for $k \in \mathbb{N}$, the Gilbert curve realization will not have the ending point of the path at the appropriate location. No Hamiltonian path is possible for paths of side lengths $((2k+1), 2)$ and end points $(0, 0)$ to $(2k, 0)$. As an artifact of the algorithm, a "hook" is formed, causing the last two points to be in different locations. See Figure **??** for an example configuration.

The hook formed for $((2k+1), 2)$ configurations can be removed for a diagonal move by swapping the last two endpoints. For the sake of clarity and brevity, we leave the algorithm as is, without adding extra complication for this

[3] https://github.com/jakubcerveny/gilbert

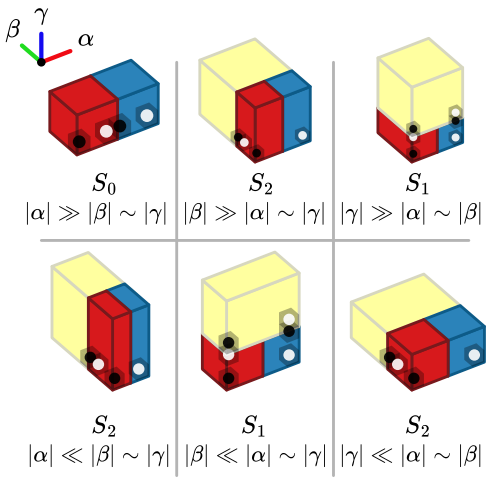

Figure 7: Eccentric cases for recursion. When the cuboid is too far from being cube-like, an S-split shape template is used to try and subdivide the cuboids into elements that are more cube like. Here α, β, γ are the width-like, height-like and depth-like dimensions. See the text or the algorithm description for the ratios used to determine eccentricity.

special case.

As mentioned in the previous subsection, when the side lengths are $(2, (2k+1))$, a long "U" shape is formed, with two long straight paths running in parallel. These pathological cases are only seen when side lengths get near their base case lengths while the other side lengths are large, comparatively. For side lengths that are larger and relatively equal in magnitude, features such as these become increasingly rare.

4. Experiments and Results

Figure 4 and Figure 8 show examples of Gilbert curves for 2D and 3D respectively.

We would like to highlight the ability of the the Gilbert curve to preserve locality analogous to the the Hilbert curve. To do this, we create plots of average distance in embedded 2D and 3D dimension relative to the linear distance along the curve. To allow for plot comparison, linear curve distances are normalized, the cumulative sum of average embedded distance is used which is then modulated by the scaling function $x^{-(1+\frac{1}{d})}$, for dimensions $d \in \{2, 3\}$. We use the term *data collapse* for the process of normalizing the x field and applying the scaling function.

Algorithm 3 gives pseudo code detailing this process. Algorithm 3 first calculates the sum of all embedded Euclidean distances for a linear curve distance, B, as well as the frequency of of each linear curve distance, M. The sum, B, is averaged using M to create the average embedded Euclidean distance, S, for a given linear length along the curve and the function G is calculated by normalizing the x value and applying the scaling function $x^{-(1+1/d)}$.

Various choices of width, height and depth were chosen for 2D and 3D Gilbert curves and Algorithm 3 was run on each. Figures 9 and 10 show the overlaid plots after the data collapse process, showing good agreement to the Hilbert curve.

The flat, mostly linear region from $x \in (0, 0.1]$ indicates

Algorithm 3 Cumulative Binning and Data Collapse

Create curve in dimension d, $N = |\alpha| \cdot |\beta| \cdot |\gamma|$
Initialize $B = \{0\}^N$, $S = \{0\}^N$, $M = \{0\}^N$
 # p_i, p_j points in embedded dimension (2D or 3D)
for all $i, j \in \{0, \ldots, N\} \rightarrow p_i, p_j \in \mathbb{Z}^d$ **do**
  $B_{|i-j|} \leftarrow B_{|i-j|} + |p_i - p_j|$
  $M_{|i-j|} \leftarrow M_{|i-j|} + 1$
end for
for all $k \in \{1, \ldots, N\}$ **do**
  $x \leftarrow \frac{k}{N}$
  $S_k \leftarrow S_{k-1} + \frac{B_k}{M_k}$
  $G(x) \leftarrow \frac{S_k}{x^{1+1/d}}$
end for
return G

a proper scaling function chosen. The intuition behind the relatively linear and relatively flat region is that for an index distance along the curve, this should map to an area or volume in the embedded 2D or 3D space, respectively. This means that for index position i and j with respective embedded mapped points p_i and p_j:

$$\begin{aligned} & |i-j| \propto |p_i - p_j|^d \\ \rightarrow \quad & |i-j|^{\frac{1}{d}} \propto |p_i - p_j| \end{aligned}$$

Where $|p_i - p_j|$ is the standard Euclidean norm. Replacing the above with an inequality leads to a *Hölder condition*.

Modulating the average embedded distance by the scaling function effectively normalizes the value by how space filling we would expect it to be. Since we take the cumulative distance, this adds one in the exponent for a scaling factor of $x^{-(1+1/d)}$.

Fluctuations away from the scaling factor indicate a failure for the curve to be space filling. Significant fluctuations can be seen near the ends of the plots when the binning technique succumbs to finite-size scaling effects. For Figures 9 and 10 this happens when the distances are close to the base recursion level or when the index distances are significantly proportional to the size of the curve. For example, when $x > 0.5$, the path length is larger than half the total path size and smaller scale locality properties are not expected to hold.

Though the Hilbert and Gilbert curve are space filling, there are minor fluctuations in the flat region. These fluctuations are highlighted in Figure 11, which is an enlarged view of Figure 10 restricted to the region $x \in [0.0001, 0.7]$ and $y \in [1.18, 1.45]$.

We note that the data collapse plot for the Hilbert curve has ripples, which we suspect is an artifact of the square recursive procedure used in the Hilbert curves construction. Other than to make a note of it, we don't investigate this feature, or their analogues for the Gilbert curve, further in this paper.

5. Conclusion

The Gilbert2D and Gilbert3D provide a generalization of the Hilbert curve to arbitrary rectangular regions in to two and three dimensions, respectively. Previous work had side length restrictions, involved complicated algorithms or lacked easily accessible reference implementations.

Our implementation is conceptually simple, allowing for ease of implementation. We also provide a libre/free ref-

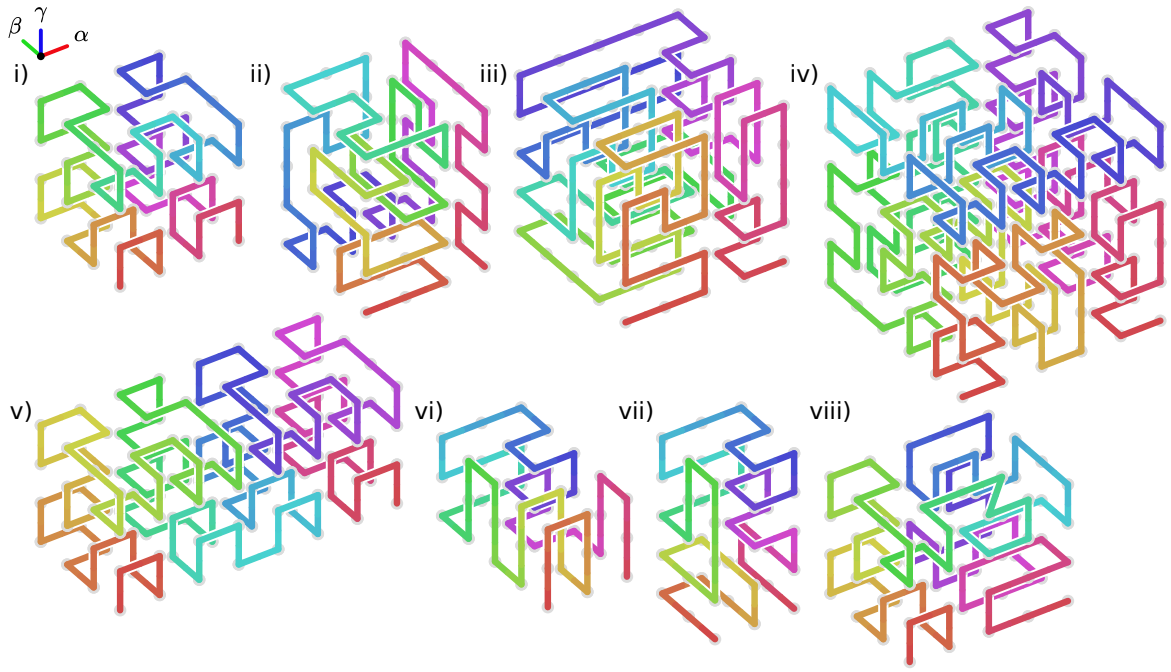

Figure 8: Examples of 3D Gilbert curves highlighting different aspects of the algorithm. i) $(4 \times 4 \times 4)$, equivalent to the Hilbert curve of the same side lengths. ii) $(4 \times 4 \times 5)$ show a J_1 split subdivision scheme. iii) $(5 \times 5 \times 5)$, shows a J_2 split subdivision scheme. iv) $(6 \times 6 \times 6)$, shows a J_0 subdivision scheme. v) $(8 \times 4 \times 4)$, shows an S_0 subdivision scheme, splitting on the α dimension. vi) $(3 \times 5 \times 3)$, shows an S_2 subdivision scheme. vii) $(3 \times 3 \times 5)$, shows an S_1 subdivision scheme. viii) $(5 \times 4 \times 4)$, shows an example of a configuration where a diagonal move (*notch*) is formed as there's no Hamiltonian path possible for the given endpoints.

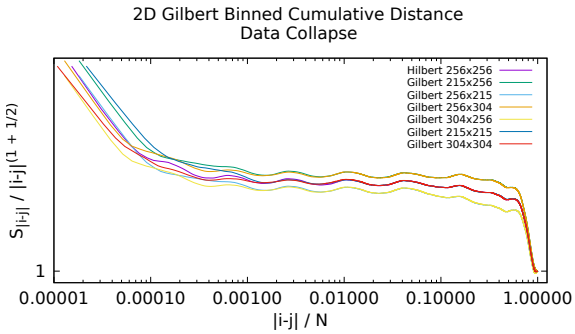

Figure 9: The data collapse for the 2D Hilbert curve of 256x256 and the Gilbert curve of combinations for $((215, 256, 304) \times (215, 256, 304))$. Note that the plot is on a log-log scale. The middle flat region indicates the scaling function of $|i-j|^{-(1+1/2)}$ is chosen correctly. The upward bend as the plots approach $x = 0$ and the dip as the plots approach $x = 1$ indicate finite-size scaling effects where the distances hit the base recursion level or have lengths that span a significant portion of the whole curve. See text for more details.

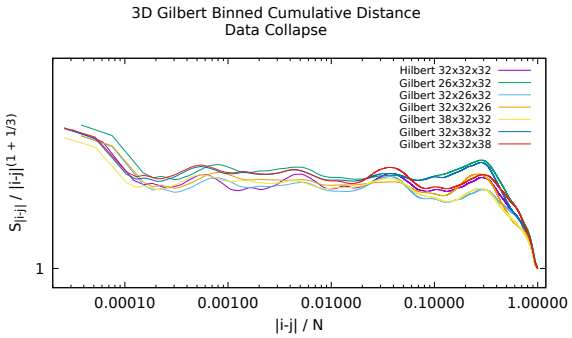

Figure 10: The data collapse for the 3D Hilbert curve of 32x32x32 and the Gilbert curve of combinations for $((26, 32, 38) \times (26, 32, 38) \times (26, 32, 38))$. Note that the plot is on a log-log scale. The middle flat region indicates the scaling function of $|i-j|^{-(1+1/3)}$ is chosen correctly. The upward bend as the plots approach $x = 0$ and the dip as the plots approach $x = 1$ indicate finite-size scaling effects where the distances hit the base recursion level or have lengths that span a significant portion of the whole curve. See text for more details.

erence implementation which can be downloaded from its repository [4].

The 2D and 3D Gilbert curve have similar locality metrics to the Hilbert curve, and provides adequate run-time and memory performance ($O(\lg N)$) for enumeration and random lookup. When a strict Hamiltonian path is not possible within the desired cuboid volume, the Gilbert curve provides a "best-effort" curve, restricting the inadmissible region to a single notch point.

Our recursive implementation provides a proof-of-concept for the method. Further Attempts at memory reductions and constant run-time execution speed optimizations are not pursued in this paper. There is also a possibility that the shape template subdivision scheme could be extended to higher dimensions but this idea is not pursued in this paper.

[4] https://github.com/jakubcerveny/gilbert.

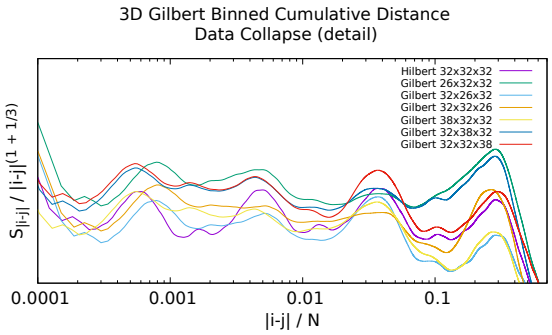

Figure 11: A detail of Figure 10 restricted to $x \in [.0001, 0.7]$ and $y \in [1.18, 1.45]$. This highlights ripples in the Hilbert curve as well as fluctuations in the Gilbert curve away from the scaling function. These fluctuations are minor and could be artifacts of the recursive cuboid subdivision process. See text for more details.

References

[1] D. Hilbert, W. Ewald, M. Hallett, U. Majer, W. Sieg, David Hilbert's lectures on the foundations of mathematics and physics, 1891-1933, number v. 5 in David Hilbert's lectures on the foundations of mathematics and physics, 1891-1933, Springer, 2004. URL: https://books.google.com/books?id=ooP9zAEACAAJ.

[2] H. Sagan, Space-Filling Curves, Universitext, Springer New York, 1994. URL: https://books.google.com/books?id=gUfvAAAAMAAJ.

[3] A. Kupers, On space-filling curves and the hahn-mazurkiewicz theorem, 2012. URL: https://api.semanticscholar.org/CorpusID:18446304.

[4] N. Wiener, The Fourier Integral and Certain of its Applications, Cambridge University Press, 1988.

[5] D. Lera, Y. Sergeyev, Lipschitz and hölder global optimization using space-filling curves, Applied Numerical Mathematics 60 (2010) 115–129. URL: https://www.sciencedirect.com/science/article/pii/S0168927409001688. doi:https://doi.org/10.1016/j.apnum.2009.10.004.

[6] T. Asano, D. Ranjan, T. Roos, E. Welzl, P. Widmayer, Space-filling curves and their use in the design of geometric data structures, Theoretical Computer Science 181 (1997) 3–15. URL: https://www.sciencedirect.com/science/article/pii/S0304397596002599. doi:https://doi.org/10.1016/S0304-3975(96)00259-9.

[7] C. Böhm, M. Perdacher, C. Plant, A novel hilbert curve for cache-locality preserving loops, IEEE Transactions on Big Data 7 (2021) 241–254. URL: https://api.semanticscholar.org/CorpusID:57781978.

[8] R. Niedermeier, K. Reinhardt, P. Sanders, Towards optimal locality in mesh-indexings, Discrete Applied Mathematics 117 (2002) 211–237. URL: https://www.sciencedirect.com/science/article/pii/S0166218X00003267. doi:https://doi.org/10.1016/S0166-218X(00)00326-7.

[9] R. Munroe, Map of the internet, 2006. URL: https://xkcd.com/195/.

[10] S. Anders, Visualization of genomic data with the hilbert curve, Bioinformatics 25 (2009) 1231–1235. URL: https://doi.org/10.1093/bioinformatics/btp152. doi:10.1093/bioinformatics/btp152.

[11] A. Cortesi, Visualizing binaries with space-filling curves, 2011. URL: https://corte.si/posts/visualisation/binvis/.

[12] R. Dafner, D. Cohen-Or, Y. Matias, Context-based space filling curves, Computer Graphics Forum 19 (2000) 209–218. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.00413. doi:https://doi.org/10.1111/1467-8659.00413.

[13] C. H. Hamilton, A. Rau-Chaplin, Compact hilbert indices: Space-filling curves for domains with unequal side lengths, Information Processing Letters 105 (2008) 155–163. URL: https://www.sciencedirect.com/science/article/pii/S0020019007002153. doi:https://doi.org/10.1016/j.ipl.2007.08.034.

[14] Y. Rong, X. Zhang, J. Lin, Modified hilbert curve for rectangles and cuboids and its application in entropy coding for image and video compression, Entropy 23 (2021). URL: https://www.mdpi.com/1099-4300/23/7/836. doi:10.3390/e23070836.

[15] L. Tautenhahn, draw a space-filling curve of arbitrary size, 2003. URL: https://lutanho.net/pic2html/draw_sfc.html.

[16] H. Haverkort, An inventory of three-dimensional hilbert space-filling curves, 2016. URL: https://arxiv.org/abs/1109.2323. arXiv:1109.2323.

[17] H. Haverkort, How many three-dimensional hilbert curves are there? (2016). doi:10.48550/arXiv.1610.00155.

[18] M. F. Mokbel, W. G. Aref, I. Kamel, Performance of multi-dimensional space-filling curves, in: Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems, GIS '02, Association for Computing Machinery, New York, NY, USA, 2002, p. 149–154. URL: https://doi.org/10.1145/585147.585179. doi:10.1145/585147.585179.

[19] C. Gotsman, M. Lindenbaum, On the metric properties of discrete space-filling curves, IEEE Transactions on Image Processing 5 (1996) 794–797. doi:10.1109/83.499920.

[20] H. Haverkort, F. van Walderveen, Locality and bounding-box quality of two-dimensional space-filling curves, Computational Geometry 43 (2010) 131–147. URL: https://www.sciencedirect.com/science/article/pii/S0925772109000765. doi:https://doi.org/10.1016/j.comgeo.2009.06.002, special Issue on the 24th European Workshop on Computational Geometry (EuroCG'08).

[21] H. Haverkort, Recursive tilings and space-filling curves with little fragmentation, 2010. URL: https://arxiv.org/abs/1002.1843. arXiv:1002.1843.

[22] D. Voorhies, I.8 - space-filling curves and a measure of coherence, in: J. ARVO (Ed.), Graphics Gems II, Morgan Kaufmann, San Diego, 1991, pp. 26–30. URL: https://www.sciencedirect.com/science/article/pii/B9780080507545500189. doi:https://doi.org/10.1016/B978-0-08-050754-5.50018-9.

[23] A. Butz, Alternative algorithm for hilbert's space-filling curve, IEEE Transactions on Computers C-20 (1971) 424–426. doi:10.1109/T-C.1971.223258.

[24] D. Moore, Fast hilbert curve generation, sorting and range queries, 2016. URL: https://web.archive.org/web/20160604090636/http://www.tiac.net/~sw/2008/10/Hilbert/moore/index.html.

[25] L. JIA, B. LIANG, M. LI, Y. LIU, Y. CHEN, J. DING, Efficient 3d hilbert curve encoding and decoding algorithms, Chinese Journal of Electronics 31

(2022) 277–284. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cje.2020.00.171. doi:https://doi.org/10.1049/cje.2020.00.171.

[26] D. Holzmüller, Efficient neighbor-finding on space-filling curves, 2019. URL: https://arxiv.org/abs/1710.06384. arXiv:1710.06384.

[27] N. Madras, G. Slade, The Self-Avoiding Walk, Probability and Its Applications, Birkhäuser Boston, 2013. URL: https://books.google.com/books?id=JsoFCAAAQBAJ.

[28] K. Christensen, N. Moloney, Complexity and Criticality, Advanced physics texts, Imperial College Press, 2005. URL: https://books.google.com/books?id=bAIM1_EoQu0C.

A. Appendix

A.1. Defect

Call the *defect* a function $\lambda_d : \mathbb{N}^d \mapsto \mathbb{N}$:

$$\lambda_2(|\alpha|, |\beta|) = \frac{|\alpha| \cdot |\beta|}{\min(|\alpha|, |\beta|)^2}$$
$$\lambda_3(|\alpha|, |\beta|, |\gamma|) = \frac{|\alpha| \cdot |\beta| \cdot |\gamma|}{\min(|\alpha|, |\beta|, |\gamma|)^3}$$

If there is a disjoint subdivision of a volume V_0 to $V_1 = (V_{0,0}, V_{0,1}, \ldots, V_{0,m-1})$, $V_0 = \cup_k V_{0,k}$, define the *average defect* of the subdivided volume to be:

$$\lambda_s(V_1) = \sum_k \frac{\text{Vol}(V_{0,k})}{\text{Vol}(V_0)} \cdot \lambda(V_{0,k})$$

This weights the defect of each subdivided cuboid by its proportional volume.

The defect gives us a coarse idea of how lopsided or *eccentric* a cuboid region is. If the defect is too high, we might want to split the larger sides while keeping the smaller sides the same size.

A.2. 2D Eccentric Split Threshold

For the 2D Gilbert curve, if the α side length is significantly longer than the β side length, we want to subdivide the rectangle into two nearly equal regions and recursively find a Gilbert curve in each region. We will justify the constant $(3/2)$ as the ratio threshold to split on. That is, when $(|\alpha|/|\beta| > 3/2)$, we split the rectangle into roughly equal parts.

The defect of a rectangle of side length $|\alpha|$ and $|\beta|$ is, with $|\alpha| > |\beta|$:

$$\begin{aligned} \lambda_2(|\alpha|, |\beta|) &= |\alpha| \cdot |\beta|/|\beta|^2 \\ &= |\alpha|/|\beta| \end{aligned}$$

After a subdivision, if we assume $|\alpha| < 2|\beta|$, the defect is:

$$\begin{aligned} \lambda_2(|\alpha|/2, |\beta|) &= \tfrac{|\alpha|}{2} \cdot |\beta|/(\tfrac{|\alpha|}{2})^2 \\ &= 2|\beta|/|\alpha| \end{aligned}$$

We're looking for the condition when there's a defect reduction, so

$$\begin{aligned} & \lambda_2(|\alpha|/2, |\beta|) < \lambda_2(|\alpha|, |\beta|) \\ \rightarrow \quad & 2|\beta|/|\alpha| < |\alpha|/|\beta| \\ \rightarrow \quad & \sqrt{2} < |\alpha|/|\beta| \end{aligned}$$

Since $\sqrt{2} \approx 1.4142 < (3/2)$, if we choose $|\alpha|/|\beta| > (3/2)$ we can be assured a defect reduction.

A.3. 3D Eccentric Split Threshold

Calculations in this section will justify what threshold value to pick of when to choose a 3D eccentric split over a J-split. Our concern is to find a simple ratio of when each of α, β or γ are considered "small enough" or "large enough", relative to the other dimensions, to split on.

An enumeration of what conditions lead to an eccentric split are as follows:

$$\begin{aligned} &(1) \quad |\alpha| >> |\beta| \sim |\gamma| \\ &(2) \quad |\beta| >> |\alpha| \sim |\gamma| \\ &(3) \quad |\gamma| >> |\alpha| \sim |\beta| \\ &(4) \quad |\beta| << |\alpha| \sim |\gamma| \\ &(5) \quad |\alpha| << |\beta| \sim |\gamma| \\ &(6) \quad |\gamma| << |\alpha| \sim |\beta| \end{aligned}$$

A representation of the eccentric splits are enumerated in Figure 7. The eccentric S-split split differs from the J-split as it's only partitioning the cuboid along one or two dimensions instead of three.

For each of the six cases, we want to know what relative difference in sizes should be used to determine when an eccentric split should be used and how to subdivide the cuboids so as to make the resulting subdivided cuboids more cube-like.

Specifically, we want to find the ratio, σ, of when one dimension is proportionally larger or smaller, respectively, than the others. Further, we choose the ratio, ρ, of where to choose the split point of subdivision. For simplicity, we might want to subdivide at the half way point ($\rho = \frac{1}{2}$) but as we will see, this might give lopsided sub-cuboid regions and using a better split point is desirable.

In what follows, our goal is to choose a subdivision that will reduce the average defect. We assume that the start and end of the path lie in the α dimension.

Since the start and end path lie on the α dimension, we are forced to split in the α axis to subdivide the cuboid. In what follows, we always commit to splitting the α cuboid region by half, with a potentially additional step to coerce the A volume to a desired parity.

A.4. $|\alpha| >> |\beta| \sim |\gamma|$

- Shape template: S_0
- $\sigma = \frac{5}{3}$
- $\rho =$ n/a

When $|\alpha|$ is much larger than $|\beta|$ or $|\gamma|$, we split into two sub-cuboids along the α axis, which we represent as an S_0 split. As stated above, we pick the midway point for α. What's left is to decide how eccentric the cuboid should be to make the S_0 split.

Take $\sigma > 1$ and $s = \min(|\beta|, |\gamma|)$, $\sigma s = |\alpha|$. The defect of the original volume is:

$$\begin{aligned} \lambda(|\alpha|, |\beta|, |\gamma|) &= \frac{|\alpha| \cdot |\beta| \cdot |\gamma|}{\min(|\alpha|, |\beta|, |\gamma|)} \\ &= \frac{\sigma s \cdot s \cdot s}{\min(\sigma s, s, s)} \\ &= \sigma \end{aligned}$$

If $|\alpha| > 2|\beta| = 2s$, then $\min(\frac{|\alpha|}{2}, |\beta|, |\gamma|) = s$ and we have an average defect $\lambda_s(V_1) = \frac{\sigma}{2}$. Intuitively, we reduce the defect by splitting a long horizontal column into two parts, each of which is more cube like, making it more cube-like on average.

We can get a better ratio of when to split by asking what the maximum value of $|\alpha| = \sigma s$ is when there is still an average defect reduction. In this case, we restrict $|\alpha| < 2|\beta|$, with $\min(\frac{|\alpha|}{2}, |\beta|, |\gamma|) = \frac{|\alpha|}{2}$, the average defect becomes:

$$\begin{aligned}\lambda_s(V_1) &= 2(\tfrac{1}{2})\tfrac{\frac{\sigma}{2}\cdot s^3}{(\frac{\sigma}{2}s)^3} \\ &= \tfrac{4}{\sigma^2}\end{aligned}$$

We want to reduce the average defect relative to the original defect, so:

$$\begin{aligned}& \lambda_s(V_1) < \lambda(V_0) \\ \rightarrow\quad & \tfrac{4}{\sigma^2} < \sigma \\ \rightarrow\quad & 4^{\frac{1}{3}} < \sigma \\ \rightarrow\quad & 4^{\frac{1}{3}} = 1.58740\cdots < \sigma < \tfrac{5}{3}\end{aligned}$$

For any $\sigma > 4^{\frac{1}{3}}$, we know the average defect will be reduced. For algorithmic simplicity, we've chosen the simple fraction $\frac{5}{3}$ (as $\frac{5}{3} > 4^{\frac{1}{3}}$) for the eccentric S_0 split ratio.

A.5. $|\gamma| >> |\alpha| \sim |\beta|$ or $|\beta| << |\alpha| \sim |\gamma|$

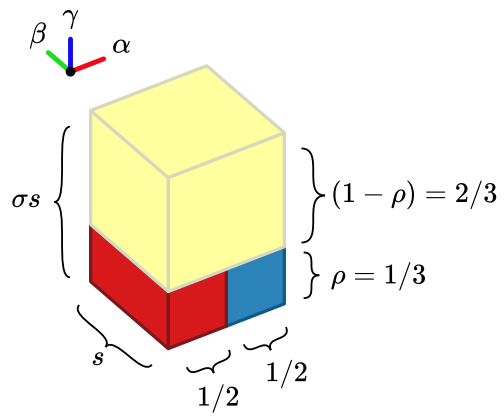

Figure 12: An S_1 eccentric subdivision showing the threshold, σ, used to determine when we should use this split and the split point ratio, ρ, to reduce the defect for a more harmonious subdivision.

- Shape template: S_1
- $\sigma = \frac{3}{2}$
- $\rho = \frac{1}{3}$

Here, we are trying to validate two parameters, σ and ρ. σ is the ratio of when $|\gamma|$ (resp. $|\beta|$) is larger (resp. smaller) to the minimum (resp. maximum) of the remaining side lengths as the threshold to trigger an S_1 template split. ρ is the proportion along the γ (resp. β) direction of where to choose the γ direction split. The S_1 shape template has three subdivided cuboids, which we call A, B and C, ordered by the path traversal through the cuboids.

We will show that there is an average defect reduction when choosing $\sigma = \frac{3}{2}$ and $\rho = \frac{1}{3}$. If we take $s = \min(|\alpha|, |\beta|)$ and $|\gamma| = \sigma s$, then the defect of the original volume is:

$$\lambda(V_0) = \sigma$$

From our choice of σ and ρ, $\min(\sigma\rho s, \frac{s}{2}) = \frac{s}{2}$, implying the minimum side length for A and C is $\frac{s}{2}$ and the minimum side length of B is $\min(\sigma(1-\rho)s, s) = s$. From this, we can calculate the average defect of the subdivided cuboid:

$$\begin{aligned}\lambda_s(V_1) &= 2\cdot(\tfrac{1}{2})\rho[\tfrac{\rho\sigma s\cdot\frac{s}{2}\cdot s}{(\frac{s}{2})^3}] \\ &\quad +(1-\rho)[\tfrac{(1-\rho)\sigma s\cdot s\cdot s}{s^3}] \\ &= 4\sigma\rho^2 + \sigma(1-\rho)^2 \\ &= 4\sigma\rho^2 + \sigma(1-\rho)^2 \\ &= \tfrac{8}{9}\sigma = \tfrac{4}{3} < \tfrac{3}{2} \\ \rightarrow\quad & \lambda_s(V_1) < \lambda(V_0)\end{aligned}$$

Showing there is an average defect reduction.

A.6. $|\beta| >> |\alpha| \sim |\gamma|$ or $|\alpha| << |\beta| \sim |\gamma|$

- Shape template: S_2
- $\sigma = \frac{3}{2}$
- $\rho = \frac{1}{3}$

As with the previous case, we are trying to validate two parameters, σ, the amount of eccentricity of one side length relative to the others, and ρ, where to subdivide the cuboid along the β axis.

Choosing $\sigma = \frac{3}{2}$ implies that the defect of the original volume is $\lambda(V_0) = \sigma = \frac{3}{2}$. Further, $\min(\sigma\rho s, \frac{s}{2}) = \frac{s}{2}$, implying the A and C volumes have minimum side length of $\frac{s}{2}$ and $\min((1-\rho)\sigma s, s) = s$, implying the minimum side length of volume C is s.

This allows us to calculate the average defect of the subdivided cuboid using an S_2 shape template:

$$\begin{aligned}\lambda_s(V_1) &= 2(\tfrac{1}{2})\rho[\tfrac{\rho\sigma s\cdot\frac{s}{2}\cdot s}{(\frac{s}{2})^3}] \\ &\quad +(1-\rho)\cdot[\tfrac{(1-\rho)\sigma s\cdot s\cdot s}{s^3}] \\ &= 4\sigma\rho^2 + \sigma(1-\rho)^2 \\ &= \tfrac{8}{9}\sigma = \tfrac{4}{3} < \tfrac{3}{2} \\ \rightarrow\quad & \lambda_s(V_1) < \lambda(V_0)\end{aligned}$$

B. Auxiliary Algorithms and Procedures

B.1. Auxiliary Functions

Auxiliary Functions 4

 # integral sign function
function SGN($w \in \mathbb{Z}$)
  if $(w < 0)$ **return** -1
  if $(w > 0)$ **return** 1
  return 0
end function

 # vector integer round down divide
function DIV($v \in \mathbb{Z}^3$, $q \in \mathbb{Z}$)
  $u_0 = \text{sgn}(v_0)\lfloor|\frac{v_0}{q}|\rfloor$
  $u_1 = \text{sgn}(v_1)\lfloor|\frac{v_1}{q}|\rfloor$
  $u_2 = \text{sgn}(v_2)\lfloor|\frac{v_2}{q}|\rfloor$
  return (u_0, u_1, u_2)
end function

 # directional vector
function $\delta(v \in \mathbb{Z}^3)$
  return $(\text{sgn}(v_0), \text{sgn}(v_1), \text{sgn}(v_2))$
end function

 # Test if q in directional volume with origin p
function INBOUNDS($q, p, \alpha, \beta, \gamma \in \mathbb{Z}^3$)
  $v \leftarrow \alpha + \beta + \gamma$
  for $i \leftarrow 0$ **to** 2 **do**
    return *false* **if** $(\text{sgn}(v_i) \cdot (q_i - p_i) < 0)$
    return *false* **if** $(\text{sgn}(v_i) \cdot (q_i - p_i - v_i) \geq 0)$
  end for
  return *true*
end function

 # base case
function HILBERT2X2X2(p, α, β, γ)
  yield p
  yield $p + \delta(\beta)$
  yield $p + \delta(\beta) + \delta(\gamma)$
  yield $p + \delta(\gamma)$
  yield $p + \delta(\alpha) + \delta(\gamma)$
  yield $p + \delta(\alpha) + \delta(\beta) + \delta(\gamma)$
  yield $p + \delta(\alpha) + \delta(\beta)$
  yield $p + \delta(\alpha)$
end function

B.2. Gilbert3D S-Split Functions

Procedure 5 S_0-Split function (eccentric split)

 # split halfway on α
function $S_0(p, \alpha, \beta, \gamma)$
  $\alpha_2 \leftarrow \text{div}(\alpha, 2)$
  if $(|\alpha| > 2)$ and $((|\alpha_2| \bmod 2) \equiv 1)$ **then**
    $\alpha_2 \leftarrow \alpha_2 + \delta(\alpha)$
  end if

  yield Gilbert3D(p, α_2, β, γ)

  yield Gilbert3D($p + \alpha_2$, $(\alpha - \alpha_2), \beta, \gamma$)
end function

Procedure 6 S_1-Split function (eccentric split)

 # split $\frac{1}{3}$ *on* γ *and halfway on* α
function $S_1(p, \alpha, \beta, \gamma)$
  $\alpha_2, \gamma_3 \leftarrow \text{div}(\alpha, 2), \text{div}(\gamma, 3)$
  if $(|\alpha| > 2)$ and $((|\alpha_2| \bmod 2) \equiv 1)$ **then**
    $\alpha_2 \leftarrow \alpha_2 + \delta(\alpha)$
  end if
  if $(|\gamma| > 2)$ and $((|\gamma_3| \bmod 2) \equiv 1)$ **then**
    $\gamma_3 \leftarrow \gamma_3 + \delta(\gamma)$
  end if

  yield Gilbert3D(p, $\gamma_3, \alpha_2, \beta$)

  yield Gilbert3D($p + \gamma_3$, $\alpha, \beta, (\gamma - \gamma_3)$)

  yield Gilbert3D($p + \alpha - \delta(\alpha) + \gamma_3 - \delta(\gamma)$, $\gamma_3, (\alpha - \alpha_2), \beta$)
end function

Procedure 7 S_2-Split function (eccentric split)

 # split $\frac{1}{3}$ *on* β *and halfway on* α
function $S_2(p, \alpha, \beta, \gamma)$
  $\alpha_2, \beta_3 \leftarrow \text{div}(\alpha, 2), \text{div}(\beta, 3)$
  if $(|\alpha| > 2)$ and $((|\alpha_2| \bmod 2) \equiv 1)$ **then**
    $\alpha_2 \leftarrow \alpha_2 + \delta(\alpha)$
  end if
  if $(|\beta| > 2)$ and $((|\beta_3| \bmod 2) \equiv 1)$ **then**
    $\beta_3 \leftarrow \beta_3 + \delta(\beta)$
  end if

  yield Gilbert3D(p, $\beta_3, \gamma, \alpha_2$)

  yield Gilbert3D($p + \beta_3$, $\alpha, (\beta - \beta_3), \gamma$)

  yield Gilbert3D($p + \alpha - \delta(\alpha) + \beta_3 - \delta(\beta)$, $-\beta_3, \gamma, -\alpha$)
end function

B.3. Gilbert3D J-Split Functions

Procedure 8 J_0-Split function

 # $|\gamma|$ even
function $J_0(p, \alpha, \beta, \gamma)$

$\alpha_2, \beta_2, \gamma_2 \leftarrow \text{div}(\alpha, 2), \text{div}(\beta, 2), \text{div}(\gamma, 2)$

 # prefer initial block even
$\alpha_2 = \alpha_2 + \delta(\alpha)$ **if** $(|\alpha| > 2)\text{and}(|\alpha_2| \bmod 2 \equiv 1)$
$\beta_2 = \beta_2 + \delta(\beta)$ **if** $(|\beta| > 2)\text{and}(|\beta_2| \bmod 2 \equiv 1)$
$\gamma_2 = \gamma_2 + \delta(\gamma)$ **if** $(|\gamma| > 2)\text{and}(|\gamma_2| \bmod 2 \equiv 1)$

yield Gilbert3D(p, $\beta_2, \gamma_2, \alpha_2$)

yield Gilbert3D($p + \beta_2$, $\gamma, \alpha_2, \beta - \beta_2$)

yield Gilbert3D($p + \beta_2 - \delta(\beta) + \gamma - \delta(\gamma)$, $\alpha, -\beta_2, -(\gamma - \gamma_2)$)

yield Gilbert3D($p + \alpha - \delta(\alpha) + \beta_2 + \gamma - \delta(\gamma)$, $-\gamma, -(\alpha - \alpha_2), (\beta - \beta_2)$)

yield Gilbert3D($p + \alpha - \delta(\alpha) + \beta_2 - \delta(\beta)$, $-\beta_2, \gamma_2, -(\alpha - \alpha_2)$)
end function

Procedure 9 J_1-Split function

 # $|\gamma|$ odd, one of $|\alpha|$ or $|\beta|$ even
function $J_1(p, \alpha, \beta, \gamma)$

$\alpha_2, \beta_2, \gamma_2 \leftarrow \text{div}(\alpha, 2), \text{div}(\beta, 2), \text{div}(\gamma, 2)$

 # prefer β_2, γ_2 even but force α_2 odd
$\alpha_2 = \alpha_2 + \delta(\alpha)$ **if** $(|\alpha| > 2)\text{and}(|\alpha_2| \bmod 2 \equiv 0)$
$\beta_2 = \beta_2 + \delta(\beta)$ **if** $(|\beta| > 2)\text{and}(|\beta_2| \bmod 2 \equiv 1)$
$\gamma_2 = \gamma_2 + \delta(\gamma)$ **if** $(|\gamma| > 2)\text{and}(|\gamma_2| \bmod 2 \equiv 1)$

yield Gilbert3D(p, $\gamma_2, \alpha_2, \beta_2$)

yield Gilbert3D($p + \gamma_2$, $\beta, \gamma - \gamma_2, \alpha_2$)

yield Gilbert3D($p + \gamma_2 - \delta(\gamma) + \beta - \delta(\beta)$, $\alpha, -(\beta - \beta_2), -\gamma_2$)

yield Gilbert3D($p + \alpha - \delta(\alpha) + \beta - \delta(\beta) + \gamma_2$, $\beta, \gamma - \gamma_2, -(\alpha - \alpha_2)$)

yield Gilbert3D($p + \alpha - \delta(\alpha) + \gamma_2 - \delta(\gamma)$, $-\gamma_2, -(\alpha - \alpha_2), \beta_2$)
end function

Procedure 10 J_2-Split function

 # $|\alpha|, |\beta|, |\gamma|$ odd
function $J_2(p, \alpha, \beta, \gamma)$

$\alpha_2, \beta_2, \gamma_2 \leftarrow \text{div}(\alpha, 2), \text{div}(\beta, 2), \text{div}(\gamma, 2)$

 # prefer β_2, γ_2 even but force α_2 odd
$\alpha_2 = \alpha_2 + \delta(\alpha)$ **if** $(|\alpha| > 2)\text{and}(|\alpha_2| \bmod 2 \equiv 0)$
$\beta_2 = \beta_2 + \delta(\beta)$ **if** $(|\beta| > 2)\text{and}(|\beta_2| \bmod 2 \equiv 1)$
$\gamma_2 = \gamma_2 + \delta(\gamma)$ **if** $(|\gamma| > 2)\text{and}(|\gamma_2| \bmod 2 \equiv 1)$

yield Gilbert3D(p, $\beta_2, \gamma, \alpha_2$)

yield Gilbert3D($p + \beta_2$, $\gamma_2, \alpha, (\beta - \beta_2)$)

yield Gilbert3D($p + \beta_2 + \gamma_2$, $\alpha, (\beta - \beta_2), (\gamma - \gamma_2)$)

yield Gilbert3D($p + \alpha - \delta(\alpha) + \beta_2 - \delta(\beta) + \gamma_2$, $-\beta_2, (\gamma - \gamma_2), -(\alpha - \delta(\alpha))$)

yield Gilbert3D($p + \alpha - \delta(\alpha) + \gamma_2 - \delta(\gamma)$, $-\gamma_2, -(\alpha - \alpha_2), \beta_2$)
end function

B.4. Gilbert2D Lookup Functions

Algorithm 11 2D Generalized Hilbert Index to Position Lookup Function (Gilbert2D_d2xyz)

```
# d_dst, d_cur ∈ ℤ, p, α, β ∈ ℤ^3
function GILBERT2D_D2XYZ(d_dst, d_cur, p, α, β)

    α_2, β_2 = div(α, 2), div(β, 2)

    if (|β| ≡ 1) then
        return p + (d_dst − d_cur) · δ(α)
    else if (|α| ≡ 1) then
        return p + (d_dst − d_cur) · δ(β)

    else if (2|α| > 3|β|) then
        if (|α_2| > 2) and (|α_2| mod 2 ≡ 1) then
            α_2 ← α_2 + δ(α)
        end if

        d_nxt ← d_cur + |α_2| · |β|
        if d_cur ≤ d_dst < d_nxt then
            return Gilbert2D_d2xyz(d_dst, d_cur,
                                   p,
                                   α_2, β)
        end if

        d_cur ← d_nxt
        return Gilbert2D_d2xyz(d_dst, d_cur,
                               p + α_2,
                               α − α_2, β)

    end if

    if (|β_2| > 2) and (|β_2| mod 2 ≡ 1) then
        β_2 ← β_2 + δ(β)
    end if

    d_nxt ← d_cur + |β_2| · |α_2|
    if d_cur ≤ d_dst < d_nxt then
        return Gilbert2D_d2xyz(d_dst, d_cur,
                               p,
                               β_2, α_2)
    end if
    d_cur ← d_nxt

    d_nxt ← d_cur + |α| · |β − β_2|
    if d_cur ≤ d_dst < d_nxt then
        return Gilbert2D_d2xyz(d_dst, d_cur,
                               p + β_2,
                               α, (β − β_2))
    end if
    d_cur ← d_nxt

    return Gilbert2D_d2xyz(d_dst, d_cur,
                           p + α − δ(α) + β_2 − δ(β),
                           β_2, −(α − α_2))
end function
```

Algorithm 12 2D Generalized Hilbert Position to Index Lookup Function (Gilbert2D_xyz2d)

```
# d_cur ∈ ℤ, q, p, α, β ∈ ℤ^3
function GILBERT2D_XYZ2D(d_cur, q, p, α, β)

    α_2, β_2 = div(α, 2), div(β, 2)

    if (|β| ≡ 1) then
        return d_cur + δ(α) · (q − p)

    else if (|α| ≡ 1) then
        return d_cur + δ(β) · (q − p)

    else if (2|α| > 3|β|) then
        if (|α_2| > 2) and (|α_2| mod 2 ≡ 1) then
            α_2 ← α_2 + δ(α)
        end if

        if inBounds(q, p, α_2, β) then
            return Gilbert2D_xyz2d(p,
                                   α_2, β)
        end if
        d_cur ← d_cur + |α_2| · |β|

        p ← p + α_2
        return Gilbert2D_xyz2d(d_cur, q, p,
                               α − α_2, β)

    end if
    if (|β_2| > 2) and (|β_2| mod 2 ≡ 1) then
        β_2 ← β_2 + δ(β)
    end if

    if inBounds(q, p, β_2, α_2) then
        return Gilbert2D_xyz2d(p,
                               β_2, α_2)
    end if
    d_cur ← d_cur + |β_2| · |α_2|

    p ← p + β_2
    if inBounds(q, p, α, (β − β_2)) then
        return Gilbert2D_xyz2d(d_cur, q, p,
                               α, (β − β_2))
    end if
    d_cur ← d_cur + |α| · |β − β_2|

    p ← p + α − δ(α) + β_2 − δ(β)
    return Gilbert2D_xyz2d(d_cur, q, p,
                           β_2, −(α − α_2))
end function
```