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 comparisons to other Hilbert curve generalization methods and investigate overall quality metrics of the Gilbert curve construction.

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
- Equivalent to the Hilbert curve when dimensions are exact powers of 2
- 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
- Similar in measures of locality to the Hilbert curve

Some drawbacks are that:

- Our implementation is recursive, which may undesirable for some applications that require better than $O(\lg N)$ runtime and memory usage
- Might not adequately capture some features of a Hilbert curve

Further, we show:

- Measures of locality are preserved (Section X)
- Trivial extensions to create generalized Moore curves (Section X)
- Comparison metrics to other space filling curves (Section X)

Generalizations of the Hilbert curve to non power of two rectangular cuboid regions has 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 local connectivity requirements preclude things like *zig-zag* Hamiltonian paths, as nearby points in the embedded dimension can be far from the origin line.

The Gilbert curve algorithm works by choosing sub rectangular cuboid regions, or blocks, to recursively find a connecting path. If one extent of the cuboid becomes too large or small relative to the other lengths, a special subdivison is performed to try 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

We concern ourselves with a space filling curve, ω, through a rectangular region (N_x, N_y, N_z), $N = (N_x \cdot N_y \cdot N_z)$:

$$\begin{aligned} k \in \quad & \{0 \dots (N-1)\}, \\ & x_k, y_k, z_k \in \mathbb{N} \\ \omega_k = \quad & (x_k, y_k, z_k) \\ \omega = \quad & (\omega_0, \omega_1, \dots, \omega_{N-1}) \\ k > 1 \quad & \rightarrow |\omega_{k-1} - \omega_k| = 1 \\ \forall i, j \in \quad & \{0 \dots (N-1)\}, i \neq j \\ \rightarrow \quad & \omega_i \neq \omega_j \end{aligned}$$

2. Related Work

3. Algorithm

3.1. Overview

For both the 2D and 3D generalized Hilbert (Gilbert) curve, a template is chosen to recursively subdivide the area or volume. When recursively descending the sub-volumes, we will need to translate to a new reference point and rotate by increments of $(\pi/2)$ radians, maintaining cuboid regions that are axis-aligned. We use the convention of labeling $\alpha, \beta, \gamma \in \mathbb{Z}^3$ for the width-like, height-like and depth-like dimension vectors, respectively.

Each of α, β and γ start off as axis-aligned vectors of integral length and are only rotated by units of $(\pi/2)$ radians, remaining axis aligned throughout. α, β and γ represent

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

the dimensions of the sub-cuboid region and the local basis when tracing out a curve.

The next sub-section discusses parity arguments for when a valid curve can be traced out in a sub-region. We then discuss the 2D Gilbert curve and end with the 3D Gilbert curve.

When talking about the 2D Gilbert curves, we assume vectors are in $\mathbb{Z}^3$ so they can be used in the more general 3D.

3.2. Valid Paths from Grid Parity

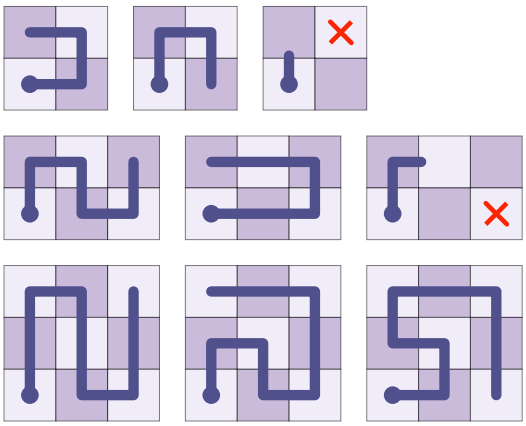

Figure 1: Illustrative examples of Hamiltonian paths height/width that are even/even, even/odd and odd/odd, respectively, when starting from the lower left hand corner

We consider a labeling of grid cell points in the volume with alternating 0 and 1 labels, representing the parity, with every axis-aligned single step move changing parity in the grid. 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. Figure 1 illustrates this for starting position $(0, 0)$ and volumes (2×2), (3×2) and (3×3), with 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 $\alpha = (w, 0, 0), \beta = (0, h, 0), \gamma = (0, 0, d)$. From this, we know that a Hamiltonian path is precluded if $((((|\beta| \cdot |\gamma|) \bmod 2) \not\equiv 1)$. That is, a Hamiltonian from p_s to p_e is precluded when α is odd and at least one of β or γ is even.

We will show that a Hamiltonian path is possible from p_s to p_e when $|\alpha|$ is even or when $|\alpha|$, $|\beta|$ and $|\gamma|$ are all odd.

3.3. 2D Generalized Hilbert Function (Gilbert2D)

For the 2D Gilbert curve, a C-split template is used, highlighted in figure 2. The C-split breaks the region into three sub-blocks, labeled A, B and C, where the A is chosen to be half the width like dimension, α, and half the height like dimension, β, with a preference to make the height like dimension, β, of even length.

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.

Algorithm 1 2D Generalized Hilbert Function (Gilbert2D)

```
# p, α, β ∈ ℤ^3
function GILBERT2D(p, α, β)

    α2, β2 = (α//2), (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
```

3.4. 3D Generalized Hilbert Function (Gilbert3D)

4. Experiments and Results

5. Conclusion

References

A. Appendix

A.1. Defect

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

$$\lambda_2(w, h) = \frac{w \cdot h}{\min(w, h)^2}$$
$$\lambda_3(w, h, d) = \frac{w \cdot h \cdot d}{\min(w, h, d)^3}$$

If there is a disjoint subdivision of a volume V_0 to $V_1 =$

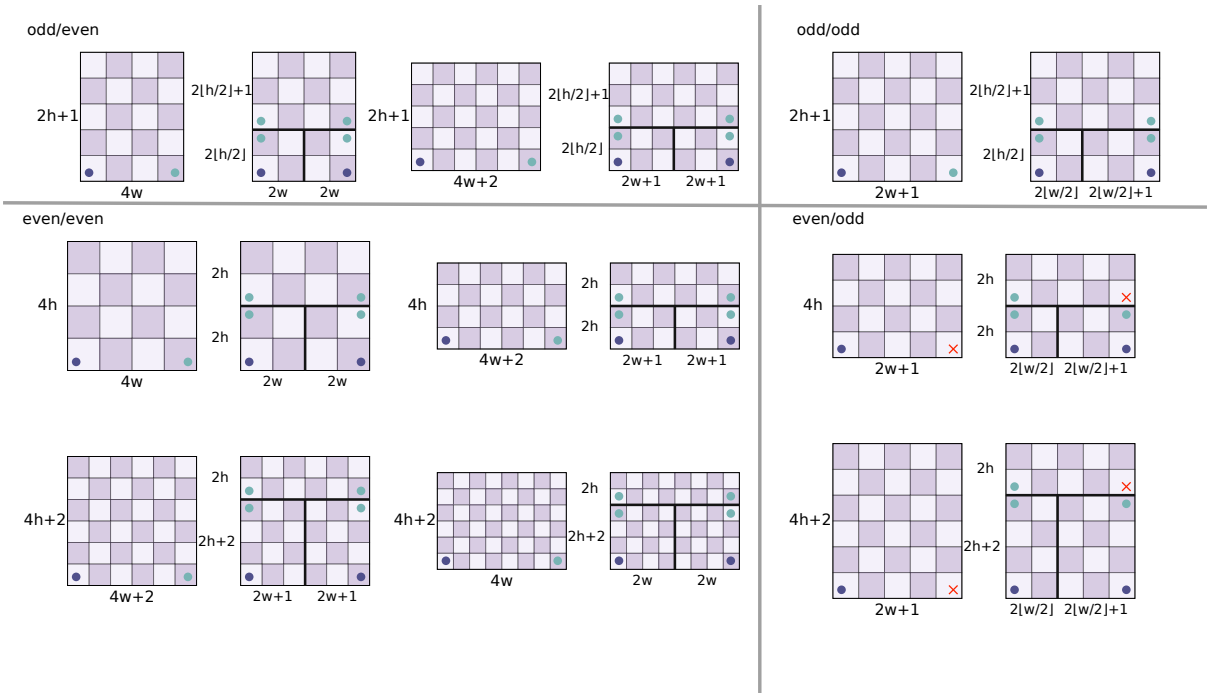

Figure 2: Enumeration of the subdivision template depending on different parities of α and β dimensions.

Procedure 2 S_0-Split function (eccentric split)

 # split halfway on α
function $S_0(p, \alpha, \beta, \gamma)$
  $\alpha_2 \leftarrow (\alpha//2)$
  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 3 S_1-Split functions (eccentric split)

 # split $\frac{1}{3}$ *on* γ *and halfway on* α
function $S_1(p, \alpha, \beta, \gamma)$
  $\alpha_2, \gamma_3 \leftarrow (\alpha//2), (\gamma//3)$
  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 4 S_2-Split function (eccentric split)

 # split $\frac{1}{3}$ *on* β *and halfway on* α
function $S_2(p, \alpha, \beta, \gamma)$
  $\alpha_2, \beta_3 \leftarrow (\alpha//2), (\beta//3)$
  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

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

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

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.

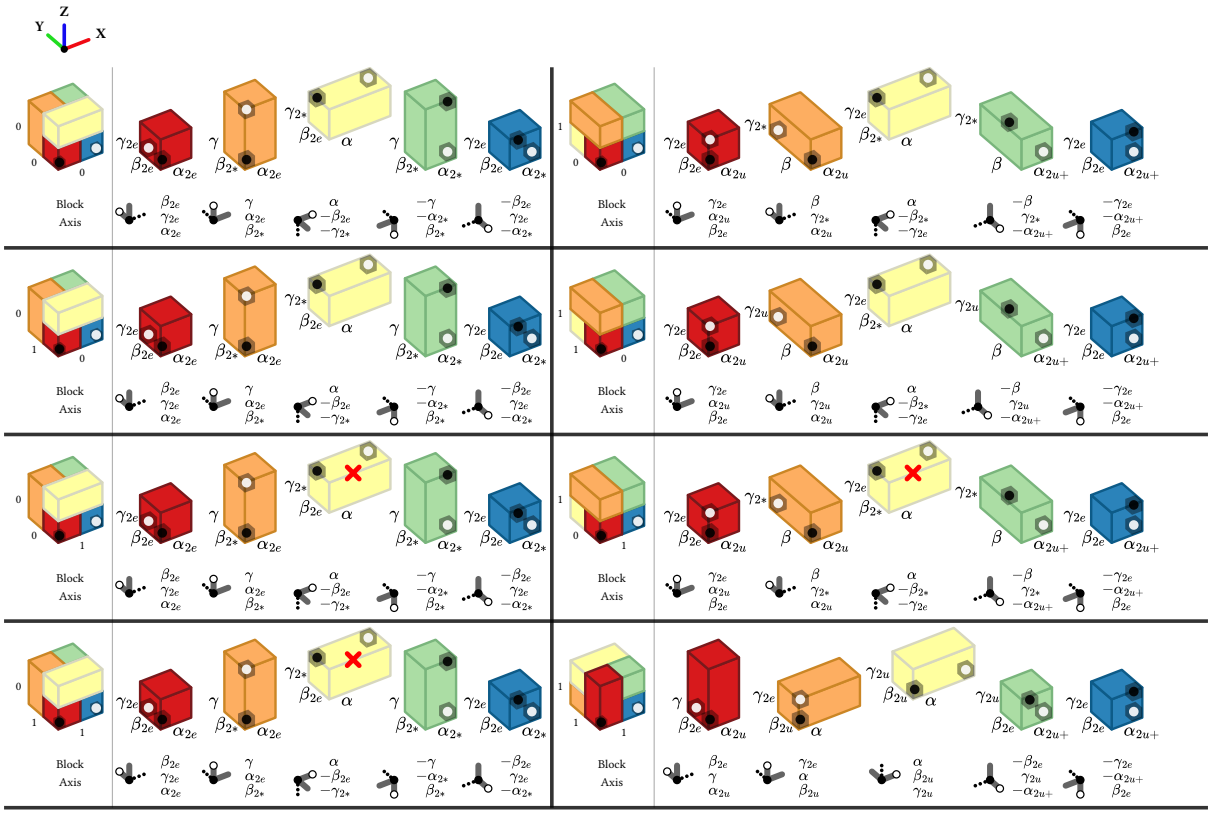

Figure 3: Bulk recursion J-split atlas for the 3D Gilbert algorithm

A.2. Eccentric Split Threshold

Calculations in this section will justify what threshold value to pick of when to choose an eccentric split over a J-split. Our concern is to find a simple ratio of when each of w, h or d 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:

$$
\begin{aligned}
w >> h \sim d (1)\\
h >> w \sim d (2)\\
d >> w \sim h (3)\\
h << w \sim d (4)\\
w << h \sim d (5)\\
d << w \sim h (6)
\end{aligned}
$$

A representation of the eccentric splits are enumerated in figure **??**. The eccentric split differs from the J-split as it's only splitting in one or two dimensions, not the full three that the J-split would be doing.

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

Specifically, we want to find the ratio, σ or η, of when one dimension is proportionally larger or smaller, respectively, than the others and 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 w dimension with the local start point at $(0, 0, 0)$ and endpoint at $(w - 1, 0, 0)$.

Because we want to avoid adding unnecessary notches, we are forced split in the w axis. We commit to always splitting the w axis in two. If we want to subdivide into three cuboid regions, we also need to pick the split point in the other dimension another dimension

A.3. $w >> h \sim d$

Take $w = \sigma s$ and $h = d = s$. We commit to splitting the width dimension in half, subdividing the original volume into two equal volumes.

The defect of the original volume is $\lambda(w, h, d) = \sigma$.

If $w > 2h = 2s$, then $\min(\frac{w}{2}, h, d) = s$ and we have an average defect $\lambda_s(V_1) = \frac{\sigma}{2}$. Intuitively, this is validation that if the width is more than twice the length of the depth and height, we make progress if we split the width in half and recursively process each sub-cuboid.

If $w < 2h$ but $\min(\frac{w}{2}, h, d) = \frac{w}{2}$, the average defect $\lambda_s(V_1) = 2(\frac{1}{2})\frac{\frac{\sigma}{2} \cdot s^3}{(\frac{\sigma}{2} s)^3} = \frac{4}{\sigma^2}$. We want to reduce the average defect relative to the original defect, so:

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

We've chosen the constant $\frac{5}{3}$ as a simple fraction above

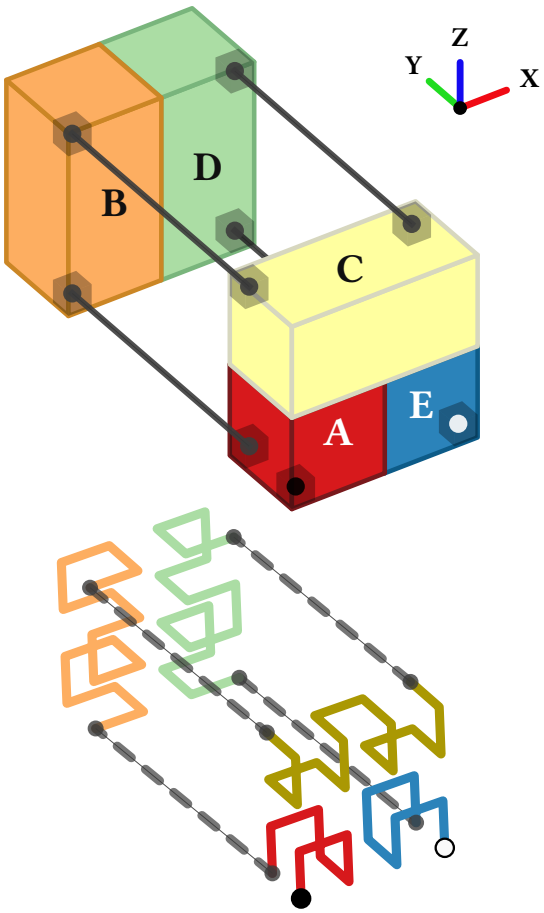

Figure 4: The J_0-split subdivision, representing the main subdivision of the bulk recursion for the 3D Gilbert curve case.

$4\frac{1}{3}$ to know when to split w.

For this case, $\sigma = \frac{5}{3}$ and $\rho = \frac{1}{2}$. That is, we split the w axis by half when the width axis exceed $\frac{5}{3}$ of both the depth, d, and height, h, dimension.

A.4. $h >> w \sim d$

Procedure 5 J_0-Split function

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

$\alpha_2, \beta_2, \gamma_2 \leftarrow (\alpha//2), (\beta//2), (\gamma//2)$

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

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 6 J_1-Split function

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

$\alpha_2, \beta_2, \gamma_2 \leftarrow (\alpha//2), (\beta//2), (\gamma//2)$

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

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

end function

Procedure 7 J_2-Split function

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

$\alpha_2, \beta_2, \gamma_2 \leftarrow (\alpha//2), (\beta//2), (\gamma//2)$

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

yield Gilbert3D(p,
β_2, γ, α_2)

yield Gilbert3D($p + \beta_2$,
γ_2, α, $(\beta - \beta_2)$)

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

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

yield Gilbert3D($p + \alpha - \delta(\alpha) + \gamma_2 - \delta(\gamma)$,
$-\gamma_2$, $-(\alpha - \alpha_2)$, β_2)
end function

Algorithm 8 3D Generalized Hilbert Function (Gilbert3D)

 # $p, \alpha, \beta, \gamma \in \mathbb{Z}^3$
function GILBERT3D(p, α, β, γ)

 # Parity of dimensions
$\alpha_0 \leftarrow (|\alpha| \bmod 2)$
$\beta_0 \leftarrow (|\beta| \bmod 2)$
$\gamma_0 \leftarrow (|\gamma| \bmod 2)$

 # Base cases
if $((|\alpha| \equiv 2)$ and $(|\beta| \equiv 2)$ and $(|\gamma| \equiv 2))$
return Hilbert3D(p,α,β,γ)
return Gilbert2D(p,β,γ) **if** $(|\alpha| \equiv 1)$
return Gilbert2D(p,α,γ) **if** $(|\beta| \equiv 1)$
return Gilbert2D(p,α,β) **if** $(|\gamma| \equiv 1)$

 # Eccentric cases
if $(3|\alpha| > 5|\beta|)$ and $(3|\alpha| > 5|\gamma|))$
return S_0(p,α,β,γ)
if $(2|\beta| > 3|\gamma|)$ or $(2|\beta| > 3|\alpha|))$
return S_2(p,α,β,γ)
if $(2|\gamma| > 3|\beta|)$
return S_1(p,α,β,γ)

 # Bulk recursion
return J_0(p,α,β,γ) **if** $(\gamma_0 \equiv 0)$
return J_1(p,α,β,γ) **if** $(\alpha_0 \equiv 0)$or$(\beta_0 \equiv 0)$
return J_2(p,α,β,γ)
end function