A Generalized 2D and 3D Hilbert Curve

Jakub Červený and 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 while preserving a notion of closeness, or locality, after the map. We present the Gilbert curve, a conceptually straight forward generalization of the Hilbert curve, that works on arbitrary rectangular regions, overcoming the limitation of the Hilbert curve that requires side lengths be exact powers of two. Our construction of the Gilbert curve introduces a notion of *harmony* that provide alternate subdivision schemes when a side length is much larger than the rest, creating more visually pleasing curves as a result.

Generalized Hilbert Curves

The discretized Hilbert curve recursively traces out a Hamiltonian path through a square region whose side length are equal and exact powers of two [**?**]. The Gilbert curve extends this idea to trace out a Hamiltonian path through to arbitrary axis-aligned cuboid regions.

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

- Conceptually simple algorithm
- Valid algorithm on arbitrary rectangular regions in 2D and 3D
- Path trace that is aesthetically pleasing

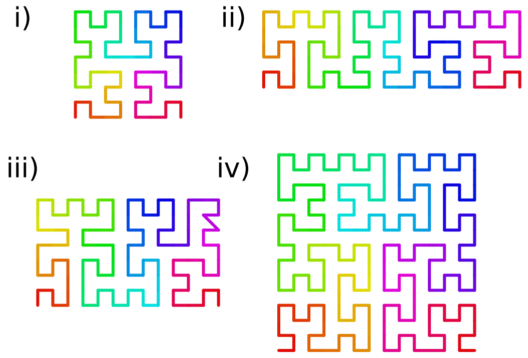

Figure 1: *2D Gilbert curves for i)* 8×8*, ii)* 18×6*, iv)* 13×8 *(with notch), iv)* 14×14

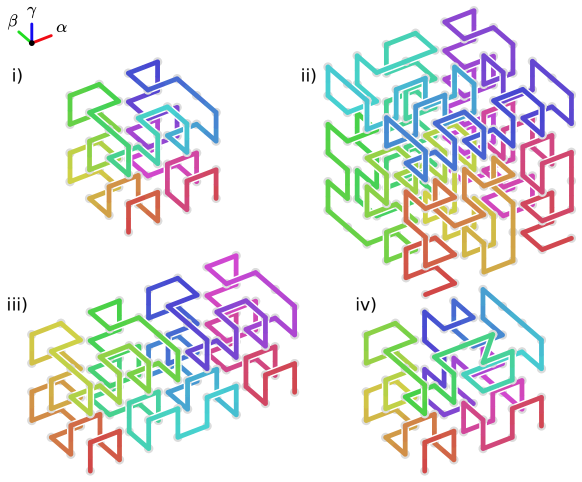

Figure 2: *3D Gilbert curves for i)* $4 \times 4 \times 4$*, ii)* $6 \times 6 \times 6$*, iii)* $8 \times 4 \times 4$*, iv)* $5 \times 4 \times 4$ *(with notch)*

In a website application and scanned note [4], 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 that help to create more *harmonious* curve realizations. Tautenhahn's exploration details the parity arguments necessary for when a subdivision scheme can be employed without creating diagonal moves.

Another method that generalizes the Hilbert curve to arbitrary rectangle regions has been done by in Zhang, Kamata and Ueshige [5] but their algorithm produces runs of straight line motifs throughout the curve. We also find their algorithm to be complex and don't see any easy way to generalize their method to 3D.

In this paper, we extend Tautenhahn's ideas to create a 2D and 3D generalized Hilbert curve. We further extend Tautenhahn's core ideas on when to use alternate subdivision schemes when one length is much larger than the rest, what we call *eccentric cases*, and apply them to a 3D generalized Hilbert curve. Tautenhahn's unpublished reasoning behind the constants used in the 2D eccentric split case are briefly discussed later [1].

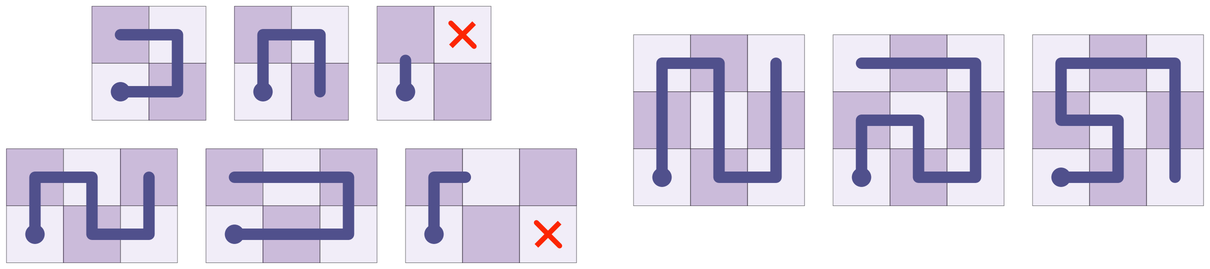

Figure 3: *Examples of Hamiltonian paths for small grid sizes. A red 'x' corresponds to no possible path for the chosen endpoints.*

Valid Paths from Grid Parity

The feasibility of determining whether there exists a non intersecting path of maximal length, called a *Hamiltonian path*, connecting endpoints on the corners 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.

Figure 3 illustrates this for starting position $(0, 0)$ with areas (2×2), (3×2) and (3×3), where a red cross indicating a precluded endpoint. When a Hamiltonian path is not precluded by the parity of the start and end point, we say that the endpoints are *color compatible*. In general, color compatibility is a necessary, but insufficient, test for a Hamiltonicity.

In the simpler case when the region is a rectangular cuboid and endpoints are on the corners of the cuboid region, color compatibility is sufficient for Hamiltonicity. This allows us a simple test to see if there is a Hamiltonian path within a rectangular region purely based on the parity of side lengths and the corner locations of the start and end points of the path. That is, when endpoints end on corners of a cuboid region, color compatibility determines if a Hamiltonian path exists.

For the constructed 2D and 3D Gilbert curves in this paper, we don't use endpoints that are diagonal within a cuboid region and only use endpoints that are in line in a single axis-aligned direction.

Subdivision Templates

For both the 2D and 3D Gilbert curve, a subdivision template is chosen to recursively partition the region. Figure 4 shows the 2D template and figure 5 provides a 2D example of the main subdivision with regions

[1]Through personal communication with the authors, Tautenhahn kindly provided the reasoning for the constants used in the eccentric split

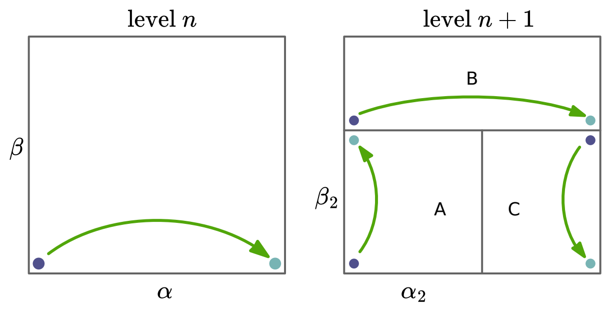

Figure 4: *Subdivision template for the 2D Gilbert curve.*

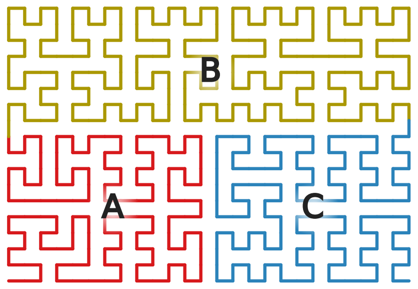

Figure 5: *Example curve with subdivision regions highlighted.*

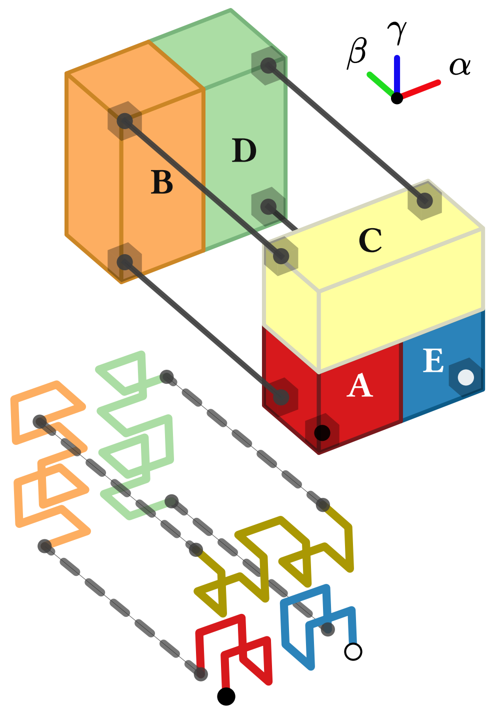

Figure 6: *The main subdivision template for the 3D Gilbert curve.*

labeled and color coded. Figure 6 shows the 3D template with a small color coded example, along with indications on how to stitch each regions endpoints to their neighbors.

The vectors $\alpha, \beta, \gamma \in \mathbb{Z}^3$ provide generalized notions of width (α), height (β), and depth (γ), so that we can transform rectangular or cuboid regions as necessary for the recursion. Each of α, β, γ will only have one non-zero component and will be orthogonal to each other, representing a snapshot of the current frame of reference.

When dividing into sub-regions, side lengths are chosen to be integral. Even side lengths are preferred for the first subdivided region (region A in figures 4 and 6). A path is recursively chosen for each of the sub-regions and, after resolution, endpoints are connected. Figure 7 shows the choice in 2D of even or odd regions depending on side length parity.

When regions have a side length that is much larger than the rest, another subdivision scheme is used. We call these cases *eccentric subdivisions*, as the larger side length makes the region lopsided.

The eccentric subdivision schemes provide a more visually pleasing partition of the space, as regimented subdivisions might take paths that are long and skinny or wide and squat. For example, partitioning a cuboid into eight roughly equal sub-regions leaves no room to absorb lopsided edge lengths. In the next section, we will briefly go over motivations for the threshold values for the 2D eccentric subdivision scheme but first provide a more detailed description of the template subdivision schemes for the 2D and 3D Gilbert curve.

2D Gilbert Curve

For the 2D Gilbert curve, the region is subdivided into two squares at the bottom and a rectangle on top, with endpoints joined on the outer perimeter. This process is recursively applied until a trivial path can be produced. Figures 4 and 10 show two different representations of the template used for the subdivision. Figures 5 and 11 show color coded examples for dimensions 26×18 and 42×30 respectively.

If the width like dimension (α) is much larger than the height like dimension (β), an alternate eccentric subdivision template is used to split the region in half along the α length (figure 8, 9).

Each regions parity is chosen so as to try and keep color compatibility, ensuring a valid path if possible. If the α axis line that the endpoints fall on is odd, with the orthogonal β direction even, a notch is forced and is directed to the upper right region, as illustrated by the “even/odd” case in figure 7.

When the side lengths are exact powers of two, the resulting curve is identical to the 2D Hilbert curve.

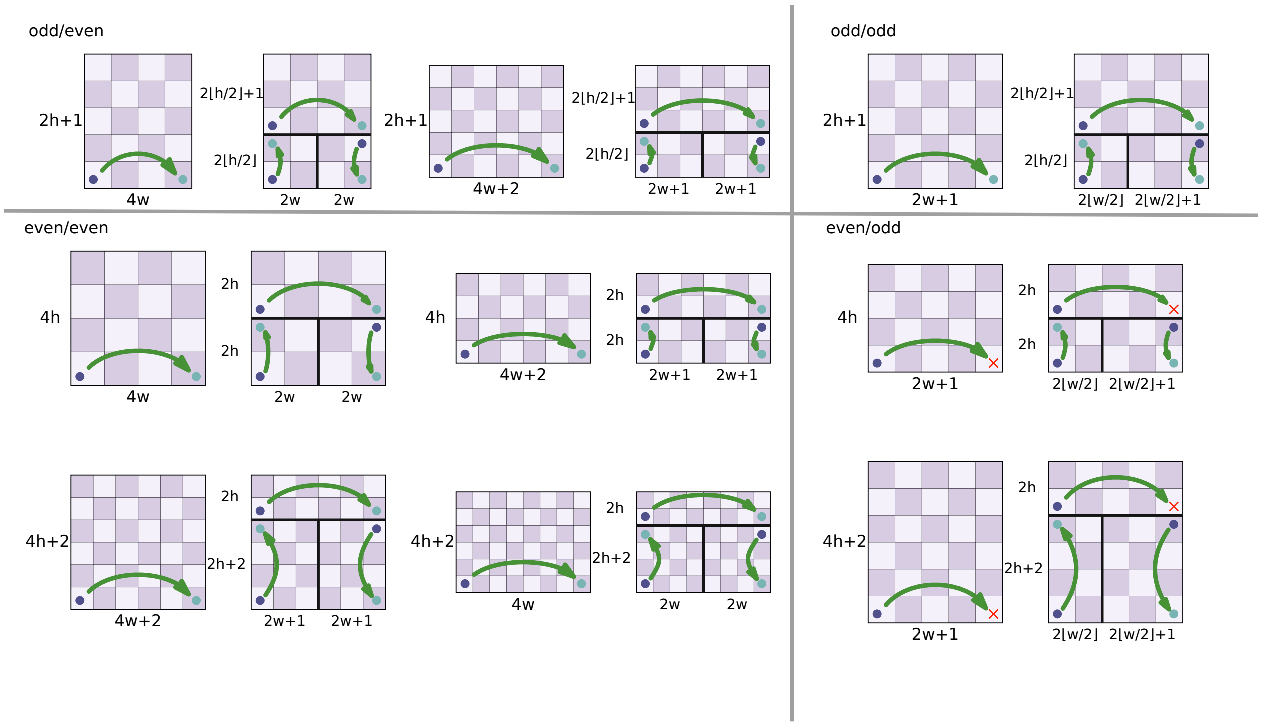

Figure 7: *Enumeration of the subdivision template depending on different parities of α and β dimensions.*

3D Gilbert Curve

In the normal case, a subdivision scheme is used to split the cuboid into five regions. Two cube like regions are partitioned where the path starts and stops, and three oblong cuboid regions are partitioned for the middle portion of the path. Figure 6 provides an exploded view of the main subdivision and where the endpoints line up. See also figure 18 for the template and figure 19 for a $10 \times 10 \times 10$ example, with each region color coded.

Endpoints within a sub-divided region are kept on the exterior of the parent cuboid and joined after the resulting recursion has completed. As with the 2D Gilbert curve, when side dimensions are equal and exact powers of two, the resulting curve is identical to the corresponding 3D Hilbert curve.

During the course of sub-division, if the width like dimension (α) is much larger, the curve is split in half along the α length (figure 12, 13). If the α eccentric split is not triggered, the height-like dimension, β, is then compared to the depth-like dimension, γ and, if very much larger, an eccentric subdivision scheme is used that splits in the α and β direction (figure 14, 15). If both α and β are not much larger, a check is done to see if the depth-like dimension, γ, is much larger than than the height-like dimension, β, splitting in the α and γ direction if so (figure 16, 17).

If any of the subdivided regions has endpoints that lay in a direction with odd length, a notch will appear. This means the number of notches for the 3D Gilbert curve isn’t limited to one in the case a side length is odd.

Algorithm 1 Gilbert 2D

```
# $p, \alpha, \beta \in \mathbb{Z}^3$
function GILBERT2D($p$, $\alpha$, $\beta$)
    $\alpha_2, \beta_2 = \mathrm{div}(\alpha, 2), \mathrm{div}(\beta, 2)$
    if ($|\beta| \equiv 1$) then
        yield $p + i \cdot \delta(\alpha)$ forall $i \in |\alpha|$
    else if ($|\alpha| \equiv 1$) then
        yield $p + i \cdot \delta(\beta)$ forall $i \in |\beta|$
    else if ($2|\alpha| > 3|\beta|$) then
        if ($|\alpha_2| > 2$) and
          ($|\alpha_2| \bmod 2 \equiv 1$) then
           $\alpha_2 \leftarrow \alpha_2 + \delta(\alpha)$
        end if
        yield Gilbert2D($p$, $\alpha_2$, $\beta$)
        yield Gilbert2D($p + \alpha_2$, $\alpha - \alpha_2$, $\beta$)
    else
        if ($|\beta_2| > 2$) and
          ($|\beta_2| \bmod 2 \equiv 1$) then
           $\beta_2 \leftarrow \beta_2 + \delta(\beta)$
        end if
        yield Gilbert2D($p$,
                        $\beta_2$, $\alpha_2$)
        yield Gilbert2D($p + \beta_2$,
                        $\alpha$, $(\beta - \beta_2)$)
        yield Gilbert2D($p + (\alpha - \delta(\alpha)) +$
                        $(\beta_2 - \delta(\beta))$,
                        $-\beta_2$, $-(\alpha - \alpha_2)$)
    end if
end function
```

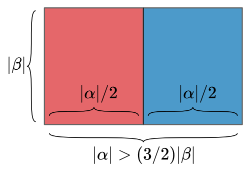

Figure 8

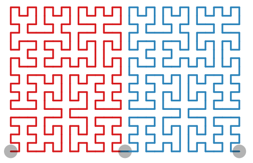

Figure 9

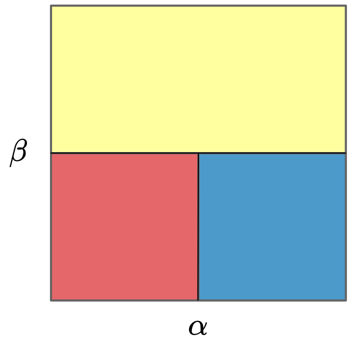

Figure 10

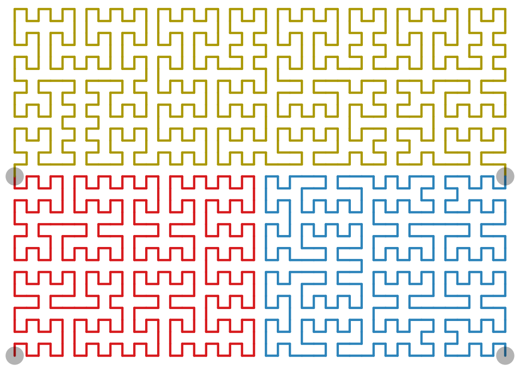

Figure 11

Harmony Through Eccentric Subdivision

When side lengths become too lopsided, an alternate *eccentric* subdivision scheme is chosen to split along the larger side length.

For each eccentric subdivision, a proportion threshold needs to be chosen to determine when a side length become too lopsided to warrant an alternate subdivision. Here we motivate the eccentric subdivision constant for the 2D case.

Consider a *defect* function, $\lambda_2 : \mathbb{N}^2 \mapsto \mathbb{N}$, that measures the area relative to what the area would be if just the minimum side length were taken:

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

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 subdivided volume's *average defect* to be the sum of defects weighted by their proportional volume:

$$\lambda_s(V_1) = \sum_k \frac{\text{Vol}(V_{0,k})}{\text{Vol}(V_0)} \cdot \lambda_2(V_{0,k})$$

The definition of the defect function is ad-hoc and has potential issues but, regardless, gives a coarse idea of how lopsided or *eccentric* a region is. If the defect is too high, we might want to split the larger sides while keeping the smaller sides the same size. Reducing the average defect attempts to make each subdivided region more square-like. When subdivided areas are more square-like, we say that the subdivided regions are more *harmonious*.

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 (see figure 8). We will justify the constant $(3/2)$ as the ratio threshold to split on, using an argument originally developed by L. Tautenhahn [2].

The defect of a rectangle with side length $|\alpha|$ and $|\beta|$, assuming $|\alpha| > |\beta|$, is $(|\alpha|/|\beta|)$. After subdivision, if we assume $|\alpha| < 2|\beta|$, the resulting defect is $2|\beta|/|\alpha|$. For a defect reduction, we search for conditions in which the ratio of the resulting defect to the original defect is less than unity, giving the equation $|\alpha|/|\beta| > \sqrt{2}$.

Since $\sqrt{2} \approx 1.4142 < (3/2)$, if we choose $|\alpha|/|\beta| > (3/2)$ we can be assured a defect reduction. In the case when $|\alpha| > 2|\beta|$, it is easy to verify that the defect is reduced, resulting in a ratio of $(1/2)$.

Defect reductions for the 3D case can be done, but are more complicated. The constants for the 3D subdivision in this paper were observed to yield visually pleasing results without concern for optimizing defect reduction.

Algorithm

Algorithm 1 and algorithm 2 shows the pseudo-code for computing the Gilbert curve in 2D and 3D respectively. After each subdivision, the origin and local axis are rotated to align with the sub region and recursively solved. In the base case where there is a unit length side, a curve in one lower dimension is output (a line for 2D and a 2D Gilbert curve for 3D). If the width like dimension, α, is oblong relative to the height-like dimension, β, an eccentric template is used. For the 3D case, additional checks to see if the remaining side lengths are oblong and, if so, the appropriate eccentric curves are applied. Otherwise, the bulk recursion is executed.

Note that even in the 2D case, α and β are taken to be vectors in 3D. 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 standard Euclidean two norm ($|v| = \sqrt{v_0^2 + v_1^2 + v_2^2}$) is used but 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. 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 $((\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1))$. The $\text{div}(a, b)$ function is used to do integer division, rounding towards zero in the case of a fractional result. To make things more concise, a convenience function is used, $(\Delta(\rho_2, \rho) \overset{\text{def}}{=} (\textit{ if } (|\rho_2| \equiv 1 \bmod 2 \text{ and } |\rho| > 2) \to \delta(\rho), \textit{ else } \to 0))$, to help coerce values to be even in the the non-degenerate case.

Random access functions to convert from index to position or position to index can be created by modifying the presented algorithms with short circuit conditionals, bypassing path calculation but keeping a running total of the path length that would have been traversed. Since each sub-region is an axis-aligned cuboid, path length within a region can easily be calculated.

Index to spatial position calculation can be done by skipping over regions that fall below the index and proceeding with the recursion for the region the index does occupy. Spatial position to index lookups can be done by finding the region the position occupies, then recursively descending, with the running index total,

[2]Given to us through personal communication with L. Tautenhahn

until the position is encountered. Both these methods then give an $O(\lg(N))$ algorithm for single index to position or position to index lookups.

Algorithm 2 Gilbert 3D

```
# $p, \alpha, \beta, \gamma \in \mathbb{Z}^3$
**function** GILBERT3D($p$, $\alpha$, $\beta$, $\gamma$)
    **return** Gilbert2D($p$,$\beta$,$\gamma$) **if** ($|\alpha| \equiv 1$)
    **return** Gilbert2D($p$,$\alpha$,$\gamma$) **if** ($|\beta| \equiv 1$)
    **return** Gilbert2D($p$,$\alpha$,$\beta$) **if** ($|\gamma| \equiv 1$)
    $\alpha_2 \leftarrow \text{div}(\alpha, 2) + \Delta(\alpha_2, \alpha)$
    $\beta_2 \leftarrow \text{div}(\beta, 2) + \Delta(\beta_2, \beta)$
    $\gamma_2 \leftarrow \text{div}(\gamma, 2) + \Delta(\gamma_2, \gamma)$
    **if** ($2|\alpha| > 3|\beta|$) and ($2|\alpha| > 3|\gamma|$))
        **yield** Gilbert3D($p$,$\alpha_2$,$\beta$,$\gamma$)
        **yield** Gilbert3D($p + \alpha_2$,$\alpha - \alpha_2$,$\beta$,$\gamma$)
    **if** ($3|\beta| > 4|\gamma|$)
        **yield** Gilbert3D($p$,$\beta_2$,$\gamma$,$\alpha_2$)
        **yield** Gilbert3D($p + \beta_2$,$\alpha$,$\beta - \beta_2$,$\gamma$)
        **yield** Gilbert3D($p+$
                        $(\alpha - \delta(\alpha))+$
                        $(\beta_2 - \delta(\beta))$,
                        $-\beta_2$,$\gamma$,$-(\alpha - \alpha_2)$)
    **if** ($3|\gamma| > 4|\beta|$)
        **yield** Gilbert3D($p$,$\gamma_2$,$\alpha_2$,$\beta$)
        **yield** Gilbert3D($p + \gamma_2$,$\alpha$, $\beta$, $\gamma - \gamma_2$)
        **yield** Gilbert3D($p+$
                        $(\alpha - \delta(\alpha))$
                        $(\gamma 2 - \delta(\gamma))$,
                         $-\gamma_2$,$-(\alpha - \alpha_2)$, $\beta$)

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

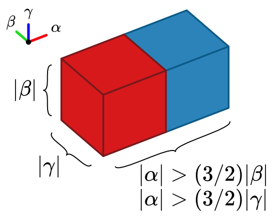

Figure 12

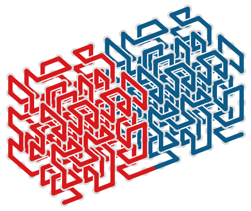

Figure 13

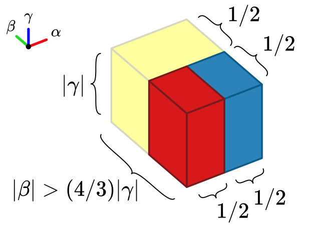

Figure 14

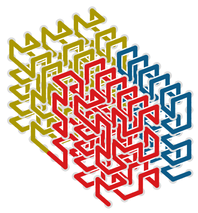

Figure 15

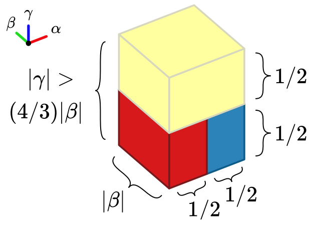

Figure 16

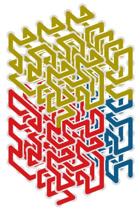

Figure 17

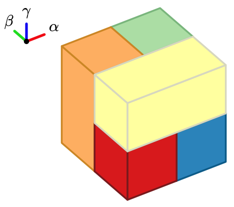

Figure 18

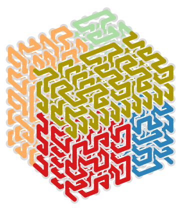

Figure 19

Conclusion and Future Directions

The Gilbert curve presented in this paper provides one interpretation for a generalized Hilbert curve in 2D and 3D to arbitrary side lengths. Generalizing a Hilbert space filling curve to arbitrary side lengths is open to interpretation and the solution presented here is one such adaptation.

The locality property of space filling curves allows for visualization of one-dimensional data to higher dimensions to keep local correlation that might be presented. This idea has been used to visualize a variety of data sets, such genomic data [1] and binary executables [2], as Hilbert curves, with a compromise that the data is padded to satisfy the power of two requirement of the Hilbert curve. The Gilbert curve offers a more immediate method to visualize data in this way and has already found uses for visualizations ([3]).

When designing an algorithm to generalize the Hilbert curve, four features that might be desirable are:

Hilbert-esque	Reduces to a Hilbert curve when side lengths are equal and are exact powers of two
Stability	Curve realizations converge as the cuboid side lengths increase proportionally
Harmony	Realizations use eccentric splits to reduce the *defect*
Notch-Limited	Curve realizations are notch free or limited to a single notch if necessary

The 2D Gilbert curve presented here has all of these properties while the 3D Gilbert curve lacks harmony and the notch-limited feature. It would be interesting to have a 3D Gilbert curve that has all of these features, perhaps by relaxing strict endpoint requirements at the sub-divided cuboid region corners. Some other ideas are to extend the concepts above to other space filling curves or circuits, or, more generally, to try and create an arbitrary space filling curve in a cuboid region with arbitrary endpoints.

A libre/free/open implementation for the Gilbert curve in 2D and 3D has been developed and can be downloaded from its repository [3].

References

[1] S. Anders. "Visualization of genomic data with the Hilbert curve." *Bioinformatics*, vol. 25, no. 10, pp. 1231-1235, 2009. https://doi.org/10.1093/bioinformatics/btp152

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

[3] S. Lee, K. Ma. "HINTs: Sensemaking on large collections of documents with **H**ypergraph vizualization and **INT**elligent agents." *IEEE Transactions on Visualization and Computer Graphics*, vol. 31, no. 9, 2025, pp. 5532-5546

[4] L. Tautenhahn. *Draw a space-filling curve of arbitrary size*, 2003. https://lutanho.net/pic2html/draw_sfc.html.

[5] J. Zhang, S. Kamata, and Y. Ueshige. "A Pseudo-Hilbert Scan Algorithm for Arbitrarily-Sized Rectangle Region." *Advances in Machine Vision, Image Processing, and Pattern Analysis*, Berlin, Heidelberg, 2006, pp. 290-299.

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