A Generalized 2D and 3D Hilbert Curve

Jakub Červený and Zzyv Zzyzek

Abstract

Hilbert curves are classic space-filling curves with strong locality properties, widely used for linearizing 2D and 3D data. In their discrete form, however, the standard Hilbert construction applies most naturally to square or cubic grids with side lengths that are exact powers of two. We present the *Gilbert curve*, a simple recursive, Hilbert-like construction that produces Hamiltonian paths on *arbitrary* 2D and *even-sized* 3D axis-aligned rectangular grids. Our construction explicitly manages endpoint parity constraints, and extends naturally to 3D.

Introduction

Space-filling curves map a one-dimensional ordering to a multi-dimensional grid while preserving locality. Discrete Hilbert curves, in particular, can be used to linearize images or volumes for cache-coherent traversal, construct spatial indexes, visualize data layout, and more [1, 2, 3, 5, 6]. In practical settings, however, underlying grids might not be square or cubic with power-of-two side lengths. Images and volumes can come in arbitrary dimensions; tiling, padding, or resampling to fit a power-of-two Hilbert curve introduces overhead and can harm locality near boundaries.

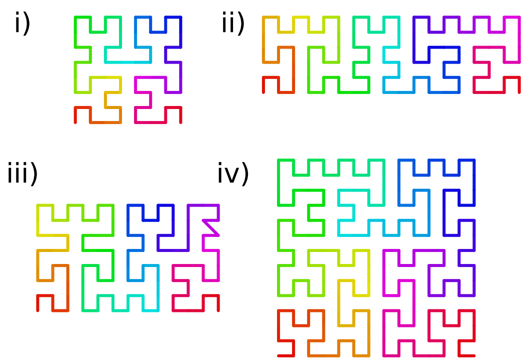


Figure 1: *2D Gilbert curves for i)* 8×8, *ii)* 18×6, *iii)* 13×8 *(with notch), iv)* 14×14.

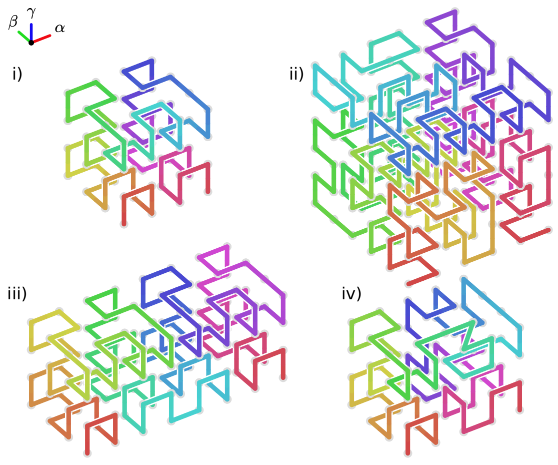


Figure 2: *3D Gilbert curves for i)* $4 \times 4 \times 4$, *ii)* $6 \times 6 \times 6$, *iii)* $8 \times 4 \times 4$, *iv)* $5 \times 4 \times 4$ *(with notch).*

This paper presents the *Gilbert curve*, a recursive construction that produces a Hamiltonian path (a non-self-intersecting path visiting every grid vertex exactly once) on arbitrary 2D rectangles and 3D cuboids. Our goal is a construction that is easy to implement, has a uniform recursive structure, and reduces to the standard Hilbert curve in the power-of-two square/cube cases. In this paper, we:

- We describe a simple recursive construction of Hilbert-like paths on arbitrary 2D rectangular grids
- We formalize a parity condition on corner endpoints and use it to guide valid recursive subdivisions
- We extend the same construction principles naturally to arbitrary 3D cuboid grids

Related Work

Hilbert introduced his eponymous curve as a continuous mapping from the unit interval to the unit square [4]. Many discrete variants exist; the most common recursive discretization applies most directly to power-of-two square (2D) and cubic (3D) grids.

Tautenhahn [7] presents an unpublished 2D construction for arbitrary sizes based on combining 3×3 Peano and 2×2 Hilbert block types. While that approach is 2D-only and not purely uniform across scales, it motivates an important endpoint-parity constraint that we make explicit.

Zhang, Kamata, and Ueshige [8] propose a pseudo-Hilbert scan for arbitrary rectangles. Compared to our construction, their method follows a more intricate case structure and does not suggest a direct extension to 3D.

Parity Constraints on Corner Endpoints

We will repeatedly need to know when a Hamiltonian path is possible between two corners of a grid. The key constraint is parity. Consider the standard checkerboard coloring of an $m \times n$ grid graph (or an $m \times n \times p$ grid graph in 3D): adjacent vertices always have opposite color. Any path alternates colors at every step, so the parity of the endpoints is constrained by the total number of visited vertices.

Lemma 1 (Color compatibility)**.** *Let G be a rectangular 2D grid graph with mn vertices, or a rectangular 3D grid graph with mnp vertices. Fix a start corner s and an end corner t. If there exists a Hamiltonian path from s to t, then s and t have the* same *color when the number of vertices is odd, and* opposite *colors when the number of vertices is even.*

Proof. Along any path, vertex colors alternate. A Hamiltonian path visits all N vertices exactly once and therefore has $N - 1$ steps. If N is odd, then $N - 1$ is even and the endpoints must have the same color; if N is even, then $N - 1$ is odd and the endpoints must have opposite colors. □

We call a pair of endpoints *color compatible* if they satisfy the lemma. Color compatibility is necessary in general; for the specific family of corner-to-corner subproblems produced by our construction, it is also sufficient, and we use it as a lightweight validity check when selecting subdivisions.

The 2D Gilbert Curve

Our 2D construction recursively partitions an $m \times n$ rectangle into a small number of axis-aligned sub-rectangles and concatenates the subpaths into a single Hamiltonian path. Figure 3 shows the main 2D subdivision template and Figure 4 illustrates one recursion with subregions highlighted.

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 subregions, we choose integral side lengths and prefer even lengths, where possible, to satisfy the parity constraints of lemma 1. Each subregion is solved recursively and then stitched to its neighbors.

When one side length is much larger than the other, we optionally apply an *eccentric split* that first bisects the long side. This improves robustness on highly elongated rectangles and keeps the recursion structure simple.

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

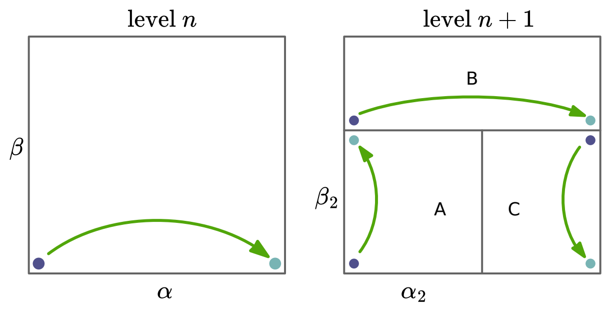


Figure 3: *Subdivision template for the 2D Gilbert curve.*

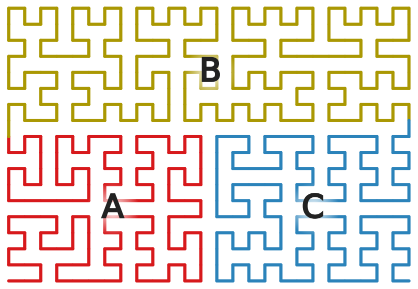


Figure 4: *Example curve with subdivision regions highlighted.*

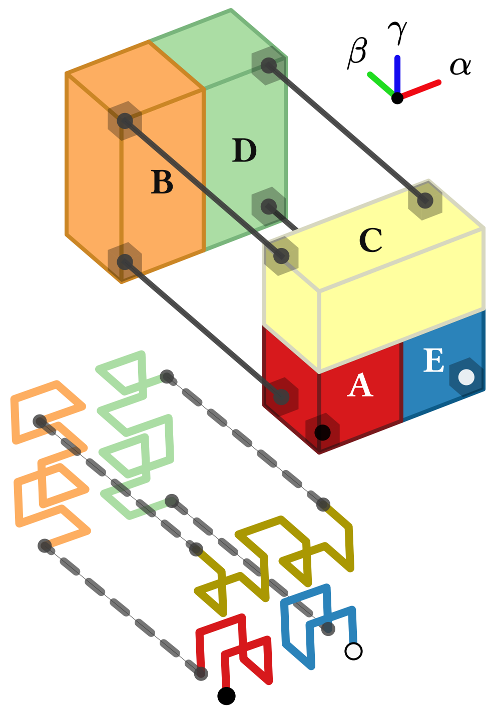


Figure 5: *The main subdivision template for the 3D Gilbert curve.*

Extension to 3D

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 5 provides an exploded view of the main subdivision and where the endpoints line up. See also figure 16 for the template and Figure 17 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 10, 11). 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 12, 13). 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 14, 15).

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.

In the case all side lengths are even, the subdivision will always choose an even length until the base case is encountered, ensuring a notch-free realization.

Algorithms

Algorithms 1 and 2 give pseudocode for the 2D and 3D constructions. We represent an oriented grid-aligned rectangle or cuboid by an origin $p \in \mathbb{Z}^3$ and axis vectors $\alpha, \beta, \gamma \in \mathbb{Z}^3$ whose nonzero components encode the side lengths and directions. The function $\delta(\cdot)$ returns the unit axis direction of its argument, and $\mathrm{div}(\cdot,\cdot)$ denotes integer division (rounding toward zero). Eccentric splits are triggered by simple aspect-ratio thresholds in the pseudocode.

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
```

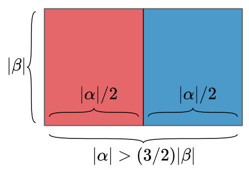


Figure 6

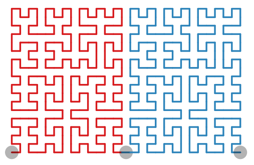


Figure 7

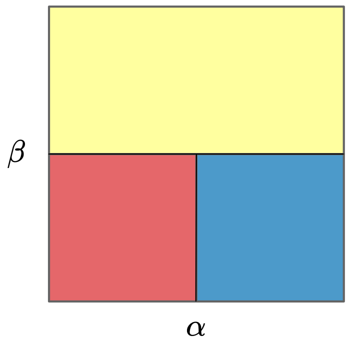


Figure 8

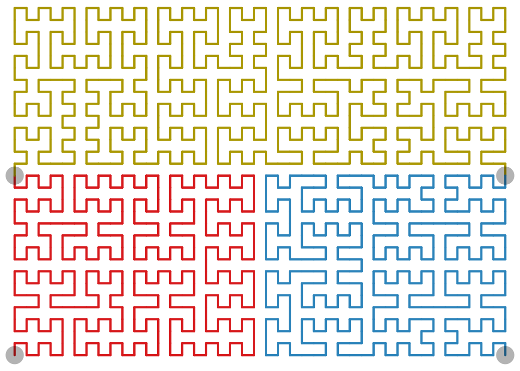
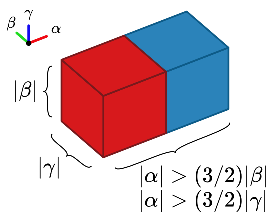


Figure 9

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($p + \alpha_2$,$\alpha - \alpha_2$,$\beta$,$\gamma$)
    else 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($p+$
                              $(\alpha - \delta(\alpha))+$
                              $(\beta_2 - \delta(\beta))$,
                              $-\beta_2$,$\gamma$,$-(\alpha - \alpha_2)$)
    else 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
```



β γ α
$|\beta|$
$|\gamma|$
$|\alpha| > (3/2)|\beta|$
$|\alpha| > (3/2)|\gamma|$

Figure 10

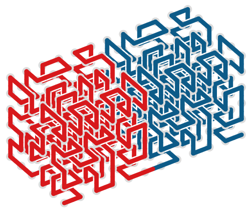


Figure 11

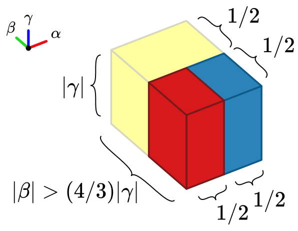


Figure 12

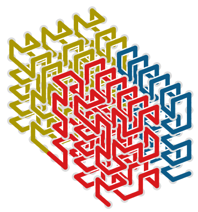


Figure 13

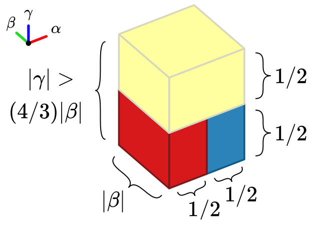


Figure 14

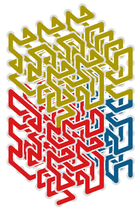


Figure 15

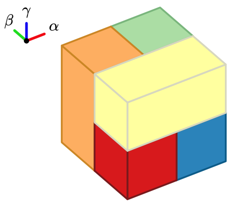


Figure 16

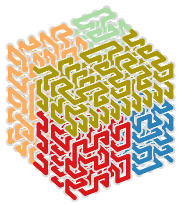


Figure 17

Summary and Conclusions

The Gilbert curve provides a practical alternative to discrete Hilbert curves when the target grid has arbitrary rectangular dimensions. This avoids padding or resampling that is otherwise required to fit power-of-two constructions, while maintaining a recursive, locality-preserving traversal. Such layouts are useful for cache-coherent image/volume traversal, spatial indexing, and visualization of 1D data on 2D/3D grids [1, 2, 3, 5].

There are several natural directions for future work. First, while our eccentric-split thresholds are simple and work well in practice, one can define more explicit optimization criteria for when and how to switch subdivision templates. Second, in 3D, relaxing subproblem endpoint constraints may allow constructions that further reduce the occurrence of forced diagonal steps when one or more side lengths are odd. Finally, more refined choices of subdivision could be explored focused on regularity metrics and/or perceptual quality.

A libre/free/open implementation for the Gilbert curve in 2D and 3D has been developed and can be downloaded from its repository [1].

Acknowledgments

AI tools were used in assisting the language and readability of this paper.

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] C. Böhm, M. Perdacher, and C. Plant "A Novel Hilbert Curve for Cache-Locality Preserving Loops." *IEEE Transactions on Big Data*, vol. 7, 2021, pp. 241-254

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

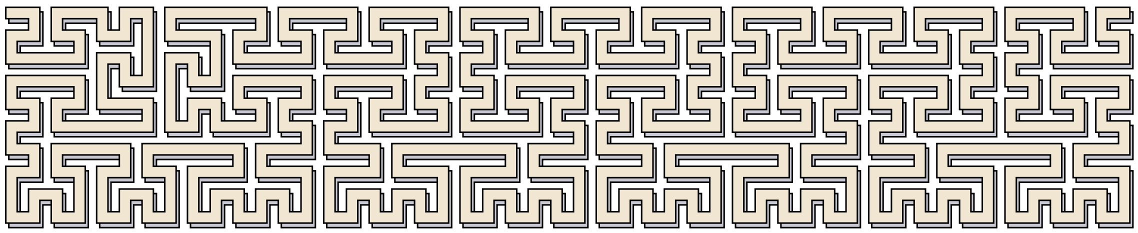
[4] D. Hilbert. "Ueber die stetige Abbildung einer Linie auf ein Flächenstück.", *Mathematische Annalen*, vol. 38, 1891, pp. 459-460

[5] B. Moon, H. Jagadish, C. Faloutsos, and J. Saltz. "Analysis of the clustering properties of the Hilbert space-filling curve." *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 1, 2001, pp. 124-141.

[6] R. Munroe. *Map of the Internet*. xkcd. 2006. https://xkcd.com/195/

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

[8] 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.



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