Published in final edited form as:

IEEE Trans Vis Comput Graph. 2021 February ; 27(2): 1591–1600. doi:10.1109/TVCG.2020.3030473.

Data-Driven Space-Filling Curves

Liang Zhou, **Chris R. Johnson**

SCI Institute, University of Utah.

Daniel Weiskopf

Visualization Research Center (VISUS), University of Stuttgart.

Abstract

We propose a data-driven space-filling curve method for 2D and 3D visualization. Our flexible curve traverses the data elements in the spatial domain in a way that the resulting linearization better preserves features in space compared to existing methods. We achieve such data coherency by calculating a Hamiltonian path that approximately minimizes an objective function that describes the similarity of data values and location coherency in a neighborhood. Our extended variant even supports multiscale data via quadtrees and octrees. Our method is useful in many areas of visualization, including multivariate or comparative visualization, ensemble visualization of 2D and 3D data on regular grids, or multiscale visual analysis of particle simulations. The effectiveness of our method is evaluated with numerical comparisons to existing techniques and through examples of ensemble and multivariate datasets.

Keywords

Space-filling curves; comparative visualization; ensemble visualization; multivariate visualization

1 Introduction

Space-filling curves (SFCs) linearize an n-D image through a one-to-one mapping into one dimension. Such linearization is useful in visualization as a tool for dimensionality reduction for 2D and 3D datasets. In this paper, we propose a data-driven space-filling curve method for data on regular or multiscale grids. Our main goal is to preserve spatial coherency (i.e., locality) and data coherency (i.e., data features) at the same time. We construct a faithful representation of the original 3D or 2D data after linearization. The method is intended for easy feature identification in the 1D visualization of space-filling curves—as line-plots—and facilitates subsequent user interactions, e.g., brushing-and-linking in comparative visualizations.

An important factor for choosing an appropriate space-filling curve is how well the locality of a dataset is preserved. Among typical space-filling curves, the Peano curve (also known as the Morton curve) [19] does not well preserve the locality, whereas the Peano-Hilbert

lzhou@sci.utah.edu.

curve [10] is considered to have better locality. Therefore, the Peano-Hilbert curve is popular in visualization. However, these space-filling curves ignore the content of a dataset.

This issue is illustrated in Fig. 1. It shows the visualization of an ensemble of a nucleon volumetric dataset generated by sampling from Gaussian distributions with varying extents of uncertainty: the volume rendering generated with a blue-white-red color map is shown in Fig. 1 (left). Boxplots along the Peano-Hilbert curve are shown in Fig. 1 (central bottom), where the feature coherency in 3D is not preserved—the small torus structure of high intensity cannot be identified. In fact, the torus is split into distant pieces in the 1D space and multiple brushes are required to select the feature (yellow areas in the “Peano-Hilbert curve” of Fig. 1). In contrast, with our method (Fig. 1 (central top)), the torus can be identified as a feature spanning a much smaller range in 1D, and can be selected with a single brush (as seen in the yellow region) thanks to better preservation of features in the spatial domain. With brushing-and-linking, the same regions are highlighted in yellow in 3D (Fig. 1 (right)) using linearizations with our method and the Peano-Hilbert curve. The better feature preservation of our method is also demonstrated with the purple brushes.

Our main contribution is a data-driven space-filling curve approach that comprises two variants of techniques: one for 2D and 3D regular grids, and another for 2D and 3D multiscale data. For regular grids, our method generates Hamiltonian cycles by replacing a minimum spanning tree using an objective function that combines locality and position terms; for multiscale data—quadtrees and octrees—our method finds adaptive Hamiltonian paths across data scales in a greedy fashion. To enable the calculation of Hamiltonian paths for multiscale data, we make a second contribution: a simple and efficient technique that finds a Hamiltonian path given only the entry and exit edges (2D) and faces (3D) of bounding rectangles/boxes of (all vertices of) grid graphs (e.g., north, east, south, west of the bounding rectangle of a 2D grid graph). We evaluate our method for each data type by comparing it to the Peano-Hilbert curve and scanline ordering on synthetic and real-world datasets. The effectiveness of our overall method is demonstrated through typical examples of 2D and 3D multivariate and ensemble data on regular grids and multiscale. The source code of our method is available online[1].

2 Related Work

Space-filling curves [24], discovered by Peano [19], are traditional topics in mathematics but now have various applications across different areas in computer science. Well-known space-filling curves include the Peano curve [19], the Gray code ordering [8], and the Peano-Hilbert curve [10]. These methods consider only spatial discretization on regular grids. Adaptively refined space-filling curves are available for multiscale data structures, specifically, quadtrees and octrees, for dynamic load balancing for high-performance computing [4]. However, these methods use static configurations that are independent of the content of the data and relate only to the size of the data.

[1] https://github.com/zhou-l/DataDrivenSpaceFillCurve.git

The context-based space-filling curve [5] is one of the few examples of a data-dependent curve. It targets to improve autocorrelation in 2D image and video encoding. There, a "dual graph" (we use this redefinition by Dafner et al. [5] throughout our paper) is generated from the input image and then a minimum spanning tree of the graph is found, where weights are determined by an objective function. Finally, the space-filling curve is constructed by replacing the minimum spanning tree with a Hamiltonian path from a Hamiltonian cycle. However, this method is limited to 2D data and does not support multiscale data, making it unsuitable for many visualization applications. Unlike this method, our data-driven space-filling curves support 3D volume data and multiscale data of 2D and 3D, which are not possible with the context-based space-filling curves [5]. In addition, our method introduces a new objective function that achieves both feature and locality coherency, making it more flexible than the context-based method. Another example is an approximation method of the space-filling curve with a data-driven metric [26]. However, only simple 2D examples with distributed points are demonstrated and it is unclear how the method could be extended to more complex data such as images and volumes. A random space-filling curve method [13] based on the "cover and merge" strategy is not data-driven but inspires the computational framework of the context-based space-filling curve [5] as well as our regular grid techniques.

Space-filling curves are useful for many visualization purposes. They facilitate comparative visualizations due to locality preservation, i.e., points that are close on the space-filling curve are close in the original 2D/3D space (not necessarily the other way around). Space-filling curves are used in ensemble visualization of 3D volumetric data [6, 28]. Peano-Hilbert curves are calculated for 3D ensemble data of multiple levels-of-details, and the linearized results are visualized as interactive enhanced line charts [6] making comparisons of 3D members possible. Similarly, a nonlinear compression method is available for the linearized 3D ensemble calculated using Peano-Hilbert curves [28]. Hilbert attention maps [16] use Peano-Hilbert curves to visualize time-varying eye-tracking data sampled on 2D regular grids as a static image, allowing features of interest that span a small neighborhood to be traced easily in the attention maps. For all methods above, brushing-and-linking is used as the major exploration approach that relates the 1D linearization and the original data. Since our technique improves space-filling curves implementations for visualization, all of these visualization applications could potentially benefit from our method.

Hamiltonian paths and cycles form the computational basis of our method. A Hamiltonian path/cycle is a path/cycle that visits each node in a graph exactly once, and a Hamiltonian cycle can be easily converted to a Hamiltonian path by a single cut on the cycle. The computation of the general Hamiltonian path problem is NP-hard [1]. For restricted scenarios, however, more efficient solutions are possible. The existence/nonexistence of a Hamiltonian path is proven for 2D grid graphs [11]; for 3D graphs of even-numbered nodes along each dimension, a Hamiltonian path can be generated from a Hamiltonian cycle [2]. However, these methods require specified entry and exit nodes, which is infeasible for data-driven space-filling curves for multiscale data. This is because if a path leaves a block of finer nodes and enters to a block of coarser nodes, we only know the exiting face of the block of finer nodes and the entering face of the block of coarser nodes. We propose a more flexible Hamiltonian path generation method—for both 2D and 3D regular grids, given

only edges/faces of entry and exit of a bounding rectangle/box—as a building block for our method.

Ensemble visualization, an active and challenging visualization topic [18], is one of the target applications of our technique. Besides the aforementioned methods using space-filling curves [6, 28], there are alternative techniques that use depth-based statistics [9, 14, 21, 29], scatterplots and parallel coordinates [23], trend graphs and parallel coordinates [17], and a flexible linked-view system with a configurable collection of statistical representations [20]. Depth-based statistics is a fundamental building block for ensemble visualization. The computation and visualization of depth-based statistics is available for 1D functions [27], 2D surfaces [9], 2D contours [29], 3D contours [21], and 2D and 3D curves [14]. In our paper, we employ a 3D extension of the surface boxplot [9] together with our data-driven curves to visualize ensemble datasets.

3 Problem Formulation

To support regular grids data and multiscale data with a unified representation, we model the input data in 2D and 3D as a graph:

$$G = (V, E, L),$$

where vertices V are nodes/vertices of the grid, edges E connect neighboring vertices (typically 4-neighbor and 6-neighbor for 2D and 3D data, respectively), and L is the level of the scale of the vertex. Our formulation facilitates a flexible multiscale representation with the per-vertex scale L, as shown in Fig. 2.

Regular grids are a special case of G where the level is constant ($L = 1$) for all vertices and the graph becomes a grid graph (Fig. 3).

Typical space-filling curves [10, 19, 24] focus on the geometry of the curves, i.e., the geometry of V and E, which concerns only the preservation of locality but not data features as the curves are ignorant of the underlying data $s(V)$. Our goal is to generate space-filling curves that preserve both locality and data features. This requires the scan order to be updated according to the data. We focus on data-driven space-filling curves that traverse through connected nodes within the graph G. Counterexamples (curves that jump between unconnected nodes) are known to have poor locality coherency, e.g., the scanline and the Peano (Morton) curve. In our case, the space-filling curve problem is equivalent to a Hamiltonian path problem [11] with coherency preservation, which allows us to formulate the generation of data-driven space-filling curves as an optimization problem of Hamiltonian paths with an objective function that takes measures of both locality preservation and data-feature preservation into account. Finding the minimum total weight of all possible Hamiltonian paths is hard on regular grids [5] and also on multiscale grids.

We denote a Hamiltonian path P through all vertices V as a sequence:

$$P = (v_1, v_2, \ldots, v_n),$$

where $v_i \in V$ is adjacent to v_{i+1} for $1 \leq i < n$. We aim to find a path $P_{\min}$ that minimizes an objective function $f(P)$:

$$P_{\min} = \arg\min_{P} f(P).$$

The objective function is formulated to be the sum of weights W that is comprised of a feature preservation term N that concerns data values $s(v)$ of vertex v, and a locality preservation term R:

$$\begin{aligned} f(P) &= \sum_{i=1}^{n-1} W(v_i, v_{i+1}), \\ W(v_i, v_{i+1}) &= (1-\alpha) N(s(v_i), s(v_{i+1})) + \alpha R(v_i, v_{i+1}), \end{aligned} \tag{1}$$

where $\alpha \in [0, 1]$ is a user-set blend factor. Our locality preservation term is a simplified, first-order locality measure. The true locality measure of space-filling curves is multiscale, and, therefore, much more complicated. However, our simplified model still yields better positional coherency compared to the scanline and the context-based method, as shown in Section 6.

Solving the minimization problem is infeasible except for extremely small datasets, and, therefore, we find an approximate optimum of the objective function. For regular grids, the optimum of $f(P)$ is approximated by adopting the strategy used by the context-based method [5] but with our new objective function and an extension to 3D. Steps involved in the framework of regular grids are illustrated in Fig. 4.

In Section 4, we briefly review the setup of the framework of the regular grid and elaborate on our new objective function and its impact. The rationale and details of this framework can be found elsewhere [5]. For multiscale data, we propose approximately minimizing the objective function $f(P)$ using a top-down and recursive greedy algorithm, which is explained in Section 5.

4 Space-Filling Curve Generation for Regular Grids

The steps for computing data-driven space-filling curves on regular grids are described in Algorithm 1. Our new contributions are a new objective function as explained in Section 4.1 and the extension to 3D detailed in Section 4.2.

We briefly review the computational framework [5] using a 2D example as illustrated in Fig. 4. For a regular grid G with an even number of vertices in each dimension, we first convert it to a graph G_c of small circuits C (Fig. 3), and then compute the dual graph G_c' (refer to the redefinition in the context-based method [5]) of G_c (Fig. 4 (a)). With the dual graph of small circuits, we are able to find $P_{\min}$ by constructing the minimum spanning tree of G_c'. The width and height of G_c' are w_d and h_d respectively; each node of G_c' corresponds to a circuit C_k for $k \in \{1, \ldots, w_d \times h_d\}$ of 2×2 vertices. The task of evaluating the weight between any vertex v_j adjacent to vertex v_i in P is now transformed to evaluating the weight on circuits

$W(C_i, C_j)$, where C_i and C_j are adjacent, and the dual of C_i is already in the minimum spanning tree (Fig. 3 (right)). A minimum spanning tree is the tree that minimizes the sum of weights among all possible trees [25], i.e., it guarantees to find $W(C_i, C_{i+1})$ as the minimum among all $W(C_i, C_j)$ in each step. The minimum spanning tree is built by joining edges of G'_c using Prim's algorithm (Fig. 4 (b)). Next, the minimum spanning tree is converted to a Hamiltonian cycle by merging the circuits according to the minimum spanning tree with the cover-and-merge strategy [13] (Fig. 4 (c)). Finally, a Hamiltonian path $P_{\min}$ is created by making a single cut anywhere in the Hamiltonian cycle [5] (Fig. 4 (d)).

Algorithm 1 Data-Driven SFC for Regular Grids

```
procedure DDSFCREGGRID(G)
    G'_c ← buildSmallCircsDualGraph(G)        ▷ G—2D/3D grid graph, G'_c—dual graph of small circuits graph of G
    W ← calculateDualGraphWeights(G'_c)       ▷ W—weights on G'_c
    MST ← findMinSpanTree(G'_c, W)            ▷ MST—minimum spanning tree
    P_min ← mergeHamCycleAndCut(MST, v_s)     ▷ v_s—entry vertex, P_min—data-driven SFC
    return P_min
```

4.1 Objective Function

In our new objective function, the definition of weights of circuits on regular grids by adding the circuit C_j to the minimum spanning tree reads:

$$W(C_i, C_j) = (1 - \alpha)N(C_i, C_j) + \alpha R(C_i, C_j), \quad \alpha \in [0, 1], \tag{2}$$

where C_i and C_j are adjacent circuits, and the dual of C_i is already in the minimum spanning tree.

For value coherency, we reuse the definition of value weights of circuits of the context-based method [5]. The value weight that grows the minimum spanning tree with C_j reads:

$$N(C_i, C_j) = |u_1| + |u_2| + |w_1| + |w_2| + |c| - |b| - |a|, \tag{3}$$

where u_1, u_2, w_1, and w_2 belong to edges along the growing direction of the minimum spanning tree, whereas a, b, and c belong to faces across the growing direction. All values above are differences of data values $s(v)$ of vertices along corresponding edges in the grid graph as defined in Fig. 3 (right).

To measure the positional coherency, we first partition the dual graph into blocks with width w_b and height h_b, and denote the block center of circuit C_k as S_{C_k}, as shown in Fig. 5. Then, we derive our positional coherency term that measures the distance of the 2D position of the circuit to the block center. The positional term is defined as follows:

$$R(C_i, C_j) = R_{\text{pos}}(C_j) = \left\| (C_j . x, C_j . y) - \left(S_{C_j} . x, S_{C_j} . y\right) \right\|, \tag{4}$$

where $R_{\text{pos}}(C_j)$ measures the positional difference as the spatial distance between C_j and the center of the block S_{C_j}. Since an edge weight is required for finding the minimum spanning

tree, we assign $R_{\text{pos}}(C_j)$ to the edge C_i–C_j in the dual graph to facilitate a unified weight definition with the value term.

A comparison of our data-driven curve for 2D regular grids and other linearization techniques is shown in Fig. 6. It can be seen that our method (Fig. 6 (Ours)) yields coherent results and correctly reveals the two peaks as coherent and neighboring features. The context-based space-filling curve (Fig. 6 (Context-based)) also reveals such structures but its spatial layout is not localized (Fig. 6 (b, Context-based)), which is confirmed by a similar autocorrelation of value (Fig. 6(d)) and a inferior autocorrelation of radial distance (Fig. 6(e)) compared to our new method). This indicates that our new method yields more coherent results than the context-based method. The scanline order (Fig. 6 (Scanline)) generates a cluttered line chart that goes up and down and it is not possible to see the data content; the Peano-Hilbert curve (Fig. 6 (Peano-Hilbert)) fails to show the two bright regions as neighboring features, and the concentrated overall structure is shown along the whole span of the line chart.

In our method, the blend factor α allows the user to flexibly change the importance of value coherency and positional coherency, which is not possible in the context-based space-filling curve [5]. Fig. 7 shows the effect of α on the traversal order of an image. The impact of α on value coherency and positional coherency is data-dependent and nonlinear. We empirically used an α value of 0.1 (except for Fig. 16, where $\alpha = 0$) as tests on datasets for evaluation (Section 6) show that such a value yields a good balance between the positional coherency and data-value coherency. We recommended using $\alpha = 0.1$ as a default and a fine-tuning of trial-and-error may be required for a specific dataset to achieve desired properties.

4.2 3D Volumes

Because a data-driven or context-based space-filling curve technique for 3D data on regular grids is useful for visualization applications [6, 28], we extend our data-driven space-filling curve to 3D regular grids. Fig. 8 shows a comparison of linearizations of a synthetic volume data—a sphere with increasing data value from exterior to interior (Fig. 8 (d))—using our data-driven method, the Peano-Hilbert curve, and scanline ordering. It can be seen that our method (Fig. 8 (a)) best preserves the value signature of the sphere as a concentrated continuous single peak with least noise, which is not possible with the Peano-Hilbert curve (Fig. 8 (b)) or the scanline (Fig. 8 (c)), the sphere is split into many pieces make the feature unidentifiable.

We extend the computational framework of the aforementioned 2D method (Algorithm 1) to 3D. Here, G_c—the equivalent to the circuit graph in 2D—is now comprised of small unit cubes—of 2×2×2 voxels—instead of circuits. The weights in the objective function have the same form as of Equation 2, but the two coherency terms are modified accordingly for 3D:

$$\begin{aligned} N(C_i, C_j) &= \sum_{r=1}^{4} (|u_r| + |w_r| + |c_r| - |b_r| - |a_r|), \\ R(C_i, C_j) &= R_{\text{pos}}(C_j), \\ R_{\text{pos}}(C_j) &= \left\| (C_j.x, C_j.y, C_j.z) - \left(S_{C_j}.x, S_{C_j}.y, S_{C_j}.z\right) \right\|. \end{aligned} \tag{5}$$

As an analogy to the 2D case, u_I and w_I are edges along the growing direction, while a_I, b_I, and c_I are faces across the growing direction. The value weights of cubes on 3D regular grids are illustrated in Fig. 9 (a). Instead of four neighbors (top, down, left, right) in the 2D case, six neighbors (with front and back as additional neighbors) are used in the 3D case when building the minimum spanning tree.

The fact that a 3D unit cube is no longer directly a cycle as in the 2D case makes the conversion from the minimum spanning tree to a Hamiltonian cycle more complicated. There exist six possible cycle configurations in a unit cube as shown in Fig. 9 (b).

After building the minimum spanning tree, we grow the Hamiltonian cycle by traversing the tree and associating unit cycles with a random configuration (from the six configurations) with association rules [2]. Two association rules for two neighboring unit cycles are adopted (Fig. 9 (c)): if parallel neighboring edges exist, we break the parallel edges and associate the four endpoints; if parallel edges do not exist, we need to break the neighboring edges and associate all eight endpoints.

Our data-driven technique could be extended for n data attributes, where multidimensional data values live in regular grids in a 2D or 3D spatial domain. The vertices in the graph now have vector-based data values, and the weights of the objective function can be defined as certain metrics of the vectors, e.g., L2-norm. Our current visualization is based on the linearization of one data attribute for multidimensional data, and one member (typically, the median) for ensemble datasets.

5 Multiscale Data-Driven Space-Filling Curves

Our data-driven curve for a multiscale data is equivalent to finding a minimum Hamiltonian path that traverses every leaf node in a quadtree or octree T. However, the aforementioned regular grid strategy is not applicable to build a dual graph for multiscale graphs because they often do not contain even-numbered vertices along each dimension (Fig. 10 (a)), and, therefore, a Hamiltonian cycle does not exist. As a result, we resort to an approximation strategy to the minimum Hamiltonian path on multiscale grids with top-down adaptive refinement.

Algorithm 2 Data-Driven SFC for Multiscale Data

procedure SFCMULTISCALE($\{I_1, I_2, \cdots, I_{L_c}\}, T$) ▷ T: tree structure (quadtree/octree)
  $P_{\text{top}} \leftarrow$ **findTopLevelSFC**(I_{L_c}) ▷ computes P_{top}—the top level SFC (data-driven)
  $P_{\text{min}} \leftarrow []$ ▷ P_{min}—SFC of the whole data
  $v_{\text{last}} \leftarrow \mathbf{0}$ ▷ v_{last}—the last SFC node
  for i in range(1, length(P_{top})) **do**
    block $\leftarrow T(P_{\text{top}}[i])$ ▷ retrieves the corresponding block of the current SFC node from the tree
    $Pz \leftarrow$ **refine**($\{I_1, I_2, \cdots, I_{L_c}\}, T$, block, v_{last}) ▷ Pz—SFC of the current block computed by adaptive refinement (data-driven)
    $P_{\text{min}} \leftarrow [P_{\text{min}}, Pz]$ ▷ appends P_{min} with Pz
    $v_{\text{last}} \leftarrow P_{\text{min}}[\text{last}]$ ▷ records the last member of P_{min}
  return P_{min}

The process of our mutliscale data-driven space-filling curve method is summarized in Algorithm 2. Given a multiscale dataset on a quadtree (Fig. 10 (a)) or octree whose nodes

are of levels $1 \leq L \leq L_c$, where 1 is for the finest level and L_c is for the coarsest level, we prepare an image/volume pyramid of L_c levels $\{I_1, I_2, \cdots, I_{L_c}\}$ for subsequent computations. First, the top-level space-filling curve P_{top} of the coarsest level I_{L_c} is found (Fig. 10 (b)). Based on the number of nodes in the coarsest pyramid level, the path is calculated using either the regular-grid-based data-driven curve method as described in Section 4 or the general Hamiltonian path method as discussed in Section 5.1. Then, we adaptively refine each element of the top-level curve P_{top} (i.e., a multiscale node in the corresponding quadtree/octree)—at each level, a minimum Hamiltonian path is found with our flexible Hamiltonian path method that improves the Hamiltonian path method on grid graphs [11, 15] (Fig. 10 (c)). Finally, the linearization is achieved: both the data value and the scale of the vertex are recorded (Fig. 10 (d)).

The objective function is approximately minimized during the process. The data value coherency term is minimized approximately with the flexible Hamiltonian path generation for each level in a block, and by finding the best entry node during adaptive refinement; the locality term is implicitly minimized by the hierarchical block-by-block advancing of the curve similar to the Peano-Hilbert curve.

In the rest of this section, we explain the flexible Hamiltonian path generation method, and then, the adaptive refinement process; finally, we discuss scenarios when the multiscale technique or the regular grids technique should be used.

5.1 Flexible Hamiltonian Path for 2D and 3D Grid Graphs

Typical Hamiltonian path methods solve the problem, i.e., (G, v_s, v_t), on a regular grid G with distinct, explicitly given entry and exit vertices v_s and v_t. However, this is not appropriate for our method as the adjacent vertices are of different scales. For example, as shown in Fig. 10, the exit vertex of the top-level block 3 (Fig. 10 (b)) cannot be known beforehand, but only the exit face of the block is known given the top-level space-filling curve. Therefore, in our formulation, we rewrite the Hamiltonian path problem as

$$(G, v_s, F_t), \tag{6}$$

where F_t is the exit side/face of the bounding rectangle/box of G. The task is then to calculate the minimum path from v_s to a valid vertex on F_t.

We have to compute the minimum path among all possible paths from entry point v_s to all valid vertices on F_t. Here, the objective function $f(P)$ is simplified to describe only the data value coherency of the sequence, and it is defined as the sum of gradient magnitude of data values $s(V)$ along the path:

$$f(P) = \sum_{i=1}^{n-1} \|s(v_{i+1}) - s(v_i)\|. \tag{7}$$

Therefore, a smoothly changing path is encouraged and a path that fluctuates significantly is punished.

We can find (or show the nonexistence of) a Hamiltonian path through given entry and exit vertices for small grids directly by using exhaustive search. A larger graph has to be partitioned into smaller ones: in practice, the largest graph that can be solved directly is 8×4 for 2D or 4×4×2 for 3D on our test machines. The limitation of the partitioning is that it may break coherent features in space, e.g., in Fig. 12, single disks/spheres are occasionally broken into different blocks and become less coherent compared to being in the same block. Therefore, we suggest as few partitions as possible if it is supported by the hardware and computational time allows. The partition is based on the relationship between the entry face and exit face of the bounding box/rectangle of the graph.

Examples of our flexible Hamiltonian path technique are shown in Fig. 11: a horizontal partitioning and a vertical partitioning of 2D graphs are shown in Fig. 11 (a) and (b), respectively; exit faces F_t on the left and top for 3D graphs are shown in (c) and (d). Since the flexible Hamiltonian path method is the building block of our multiscale space-filling curve techniques, exhaustive search is implemented in an non-recursive fashion using stacks to improve efficiency.

5.2 Adaptive Refinement

The refinement method **refine**—as described in Algorithm 3—is the core of the space-filling curve for multiscale data. If any of the nodes within the block is not a leaf node, it has to be refined all the way down to the finest level in a data-driven fashion. The key is to determine the suitable entry node for blocks at different levels (the **findBestEntry** function in Algorithm 3): we keep track of the last vertex v_{last} in the Hamiltonian path and utilize it to find the matching entry node in the next block, i.e., the node within the adjacent block to v_{last} that has the minimum difference to its data value. The combination of this process and the Hamiltonian path generation function **linearizeHamPath** (Section 5.1) approximates the minimization.

Algorithm 3 Refinement of a Multiscale Block

procedure REFINE($\{I_1, I_2, \cdots, I_{L_c}\}$, T, block, v_{last})
  $v_s \leftarrow$ **findBestEntry**(v_{last}) ▷ finds the best entry vertex v_s (data-driven)
  if needRefine(block) **then**
    $P_{\text{curr}} \leftarrow []$ ▷ P_{curr}—SFC of the current multiscale block
    $P_{\text{currTop}} \leftarrow$ **linearizeHamPath**(I_{block}, T, v_s) ▷ finds the top-level SFC P_{currTop} of the current data block I_{block} (data-driven)
    for i in range(1, length(P_{currTop})) **do**
      ctopBlock $\leftarrow$ T($P_{\text{currTop}}[i]$) ▷ ctopBlock: top-level sub-block within the current block
      if needRefine(ctopBlock) **then**
        if $i \geq 2$ **then**
          $v_{\text{last}} \leftarrow P_{\text{currTop}}[i-1]$
        $P_{\text{finer}} \leftarrow$ **refine**($\{I_1, I_2, \cdots, I_{L_c}\}$, T, ctopBlock, v_{last}) ▷ recursively computes the SFC of ctopBlock
        $P_{\text{curr}} \leftarrow [P_{\text{curr}}, P_{\text{finer}}]$ ▷ appends P_{curr} with P_{finer}
      else
        $P_{\text{curr}} \leftarrow [P_{\text{curr}}, P_{\text{currTop}}[i]]$ ▷ appends P_{curr} with $P_{\text{currTop}}[i]$
  else
    $P_{\text{curr}} \leftarrow$ **linearizeHamPath**(I_{block}, block, v_s) ▷ data-driven
  return P_{curr}

Fig. 12 (Quadtree) shows the linearization with our data-driven technique for quadtree on a synthetic image (Fig. 12 (Quadtree, first column)). The resulting multiscale linearizations and their reconstructed linearizations are shown in the second column of Fig. 12 (row 1). Here, "reconstructed" refers to generating the linearization back to the regular grid using the visit order of the coordinates of multiscale nodes and their scale information. It can be seen that our technique preserves the value signatures of five disks—each as a peak—which is more prominent in the reconstructed space-filling curve. Fig. 12 (row 1, third column) shows the geometry of the space-filling curve over the quadtree color mapped by the traversal order (from blue to yellow). An octree data of five spheres is shown in Fig. 12 (Octree): the linearization using our technique Fig. 12 (Octree, second column, top) preserves the value pattern of five spheres, more evident in the reconstructed version Fig. 12 (Octree, second column, bottom). The spatial configuration of the space-filling curve is shown in Fig. 12 (Octree, third column), with the color map showing the scan order.

6 Evaluation

Our method is evaluated by numerical comparison of autocorrelations of our techniques ($\alpha = 0.1$ for regular grid techniques) to existing linearization methods. Autocorrelation is the correlation of a signal and a shifted copy of the signal; the measurement indicates the coherency of a signal, and is suitable for measuring the effectiveness of space-filling curves [5]. In our evaluation, autocorrelations of two measures are calculated: 1) autocorrelation of linearized data values $u(i)$ that measure the data coherency of space-filling curves; 2) autocorrelation of radial Euclidean distances of elements in the linearization $t(i)$ that measures the spatial coherency, i.e., locality, of the curves. The definitions of the two measures are shown as follows:

$$u(i) = s(P(i)), \quad t(i) = \| [P(i).x, P(i).y, P(i).z] - [0,0,0] \| .$$

Note that the distance measure $t(i)$ is applicable only to regular grids and $P(i).z = 0$ for 2D cases.

We use benchmark datasets commonly employed in scientific visualization and average the autocorrelations of each dataset for each linearization technique. Specifically, 11 datasets —typically, slices of volume data (one slice each of downsampled volume datasets of aneurysm, beetle, bonsai, MRI brain, engine, foot, fuel, hurricane Isabel, neghip, and nucleon—all from a public volume data library[2]; and an image of 5 randomly placed disks)—are used in the evaluation of 2D methods, and 5 volumetric datasets (fuel, neghip, nucleon, heart ischemia, and a procedural volume generated with a tangle function; all downsampled to 32^3) are used for 3D. Autocorrelations are shown in Fig. 13, where the horizontal axis is the lag (shift) of the signal, and the vertical axis is the value of normalized autocorrelation.

Averaged autocorrelations of data values in 2D (Fig. 13 (a)) indicate that our regular grid method (green) has almost the same feature coherency as the context-based curve (purple)

[2] http://schorsch.efi.fh-nuernberg.de/data/volume/

as they are overlapping, and both perform much better than the Peno-Hilbert curve (blue) and the scanline (red); our data-driven method for quadtree (black) performs better than the Peano-Hilbert curve and scanline. In terms of averaged autocorrelations of radial distance (Fig. 13 (b)), the Peano-Hilbert curve (blue) performs best and is followed by our data-driven method (green), and then the context-based method (purple); the scanline method (red) has much worse performances than other techniques.

For 3D data, the evaluation compares our regular grid-based data-driven technique, our multiscale technique for octree, the Peano-Hilbert curve, and the scanline. As shown in Fig. 13 (c), our regular-grid method (green) tops other techniques for averaged autocorrelation of data value, and our octree technique (black) follows, and then, the Peano-Hilbert curve (blue), and the scanline (red). For autocorrelations of radial distance (Fig. 13 (d)), our regular grid method is better than the scanline but out-performed by the Peano-Hilbert curve.

The evaluation confirms that our data-driven technique balances the data value coherency and locality coherency, and is more flexible than existing techniques. The comparisons also suggest that our regular grid techniques have better data value coherency performance than our multiscale techniques—the former is preferred when high-quality linearization is needed for volumetric data and computational time is not a limiting factor.

Our multiscale techniques are most suitable for multiscale data by nature, e.g., multiscale simulation of particles. An octree can be built to have a few (even just one) particles in the finest level node so that accurate data values of almost all particles could be preserved. However, for regular grids, this is more difficult if not impossible. Either more particles are averaged out or a very fine grid has to be built. In addition, multiscale techniques are faster than regular grid techniques as fewer nodes have to be visited in multiscale structures compared to regular grids for the same input data.

In contrast, our regular grid techniques yield more coherent linearization results than our multiscale curves. This is due to that the multiscale curve uses the top-down approach to ensure a Hamiltonian path exists, but it breaks coherent features in space on certain occasions as demonstrated in Fig. 12 (Octree). The aforementioned numerical comparison of coherency confirms that the regular grid techniques have higher coherency than multiscale techniques.

Therefore, we recommend using the multiscale technique for intrinsically multiscale data, especially, point datasets, and for reducing computation time. The regular grids techniques are recommended for higher linearization quality for images and volumetric data on uniform grids. Furthermore, preprocessing the input data with a segmentation could improve the coherency, and, potentially, the efficiency of our method (for regular grids, fewer comparisons are needed when neighbors are homogeneous).

7 Visualization, User Interaction, and Implementation

Our method facilitates visualizations that use the horizontal axis for any spatial configuration along the space-filling curve, freeing the vertical space to visualize values aligned within the spatial configuration. For multivariate data, each variable can be

visualized as a line plot (Fig. 14), whereas for ensemble data, functional boxplots are used (Fig. 1, 15, 16). Here, we employ the surface boxplot [9] for 2D ensemble datasets; an extension of the method [9] to 3D is applied to 3D ensemble datasets. The conventional color scheme used for boxplots is adopted.

We have built an interactive visualization tool to support the exploration of space-filling curve-based visualization of datasets. The tool comprises three linked views: a line plot view that shows the linearized data; a 3D renderer that renders volumetric data with direct volume rendering, and particle datasets with polygon-based rendering; a 2D renderer that shows the data slice. The line plot view is linked with 2D and 3D renderers using the scan order of the space-filling curve that records the pixel ID of the line plot and its associated pixels/voxels in the original data. Brushing-and-linking allows us to brush regions in the line plot view, highlighting them in the 2D and 3D renderings. Interactive zooming and panning are supported in the line plot view so that both the overall structure and details of the line visualizations can be examined.

Our space-filling curve techniques were implemented using Matlab. The visualization tool is built using C++, Qt, and OpenGL, and is accelerated by the GPU. The QCustomPlot library [7] was used to aid the implementation of the line plot view. Our method was tested on a 2019 13-inch Macbook Pro with 2.3 GHz Intel i5 CPU, 8 GB main memory, and an Intel Iris 655 integrated GPU. The data-driven space-filling curve only needs to be computed once for a given dataset, and the computation time depends on the number of vertices in the graph representation of the dataset. Timings of generating data-driven space-filling curves for examples of the paper are summarized in Table 1. Full interactivity was achieved for the exploration of all examples.

8 Examples

We demonstrate the usefulness of our method with examples of multiscale multivariate particle data, ensemble of medical images on 2D regular grids, and volumetric ensemble datasets on regular grids. Visualizations of ensemble datasets are based on linearizations of the median members. The smooth particle hydrodynamics (SPH) dataset shown in Fig. 14 is a timestep in a dam break simulation [22]; the dataset contains particles with six attributes: density, pressure, speed, and velocity in X, Y, and Z directions, respectively. The data is decomposed into an octree and linearized using our octree-based data-driven curves. Data values of all attributes are linearized with the spatial layout of the space-filling curve of the pressure attribute. We highlight regions that are distinct from their neighborhood in the linearizations: low values of density, high values of pressure, and high values of speed and velocity. Here, the most prominent feature is the highest pressure region (brushed in purple) with values over 50,000 as shown in the zoom-in. The particles within the brushed regions can be seen in the 3D rendering (Fig. 14 (a)). Our method yields a new visual debugging method that shows clear, non-occluded quantities of each attribute embedded in a spatial context. It could complement non-spatial multivariate plots [22] for a more comprehensive visual debugging system.

Fig. 15 shows the visualization of a series of open-access MRI slices [12]. The boxplot linearized with our method (Fig. 15 (b, center)) exhibits a more coherent feature that is more concentrated than in the Peano-Hilbert curve linearization (Fig. 15 (b, bottom)). As shown in the zoom-in of Fig. 15 (b, top), the outlier (the red curve) has wider low-value regions (inside the gray boxes) than the band as shown in the zoom-in. With brushing-and-linking, it is confirmed that the outlier image (Fig. 15 (a, top)) has larger lateral ventricle area than the median image (Fig. 15 (a, bottom)) with the brush on the boxplot linearized with our data-driven space-filling curve.

The myocardial ischemia dataset was generated by image-based, experimentally derived cardiac electrical potential simulation [3]. We use a subset of ensemble runs of the simulation and sample the data on regular grids for our experiment. Here, we are interested in the acute ischemic regions associated with mean potentials greater or equal to 3 eV. As shown in Fig. 16 (b, top), the linearized 3D boxplot using data-driven technique yields more concentrated global features than the linearization with the Peano-Hilbert curve (Fig. 16 (b, bottom)). The region of interest (high potential regions) is bounded in a small neighborhood with our method that could be selected with one brush (Fig. 16 (b, top)), whereas the Peano-Hilbert curve yields a more scattered result—a large number of brushes are required (Fig. 16 (b, bottom)). The volume rendering (Fig. 16(a)) of the median ensemble member shows that the region of interest is spatially continuous (white in the rendering); the highlighted regions in space (Fig. 16 (c)) verify that our method gives good coherency of the feature.

9 Conclusion and Future Work

We have introduced data-driven space-filling curves for 2D and 3D visualization. We have designed our methods to preserve coherency of both data value and locality after the mapping from the spatial domain to 1D. The methods are applicable for data on regular grids and in multiscale. We have modeled the problem as finding a Hamiltonian path that approximates the minimum of an objective function that blends a data value term and a locality term. Two variants of techniques are available for regular grids and multiscale data (quadtrees and octrees). The effectiveness of our method has been evaluated by comparing to existing methods on various datasets with qualitative visual comparison and quantitative comparisons of autocorrelations. We have confirmed that existing methods cannot preserve both data features and locality after linearization. Through multivariate and ensemble visualization examples with a wide range of real-world datasets, we have demonstrated the usefulness of our data-driven space-filling curves.

In the future, we would like to extend our method for time-varying data to understand the coherency in time. The positional term for regular grids requires uniform blocks of a user-defined size, which should be improved to be data-driven. The method could be used for multi-field visualization such that different field data, e.g., scalar, vector, and tensor, could be visualized in a linear layout for non-occluded comparisons and investigation of correlations. Finally, we would also like to accelerate our method with parallel computing to support larger datasets.

Acknowledgments

This work is supported in part by the Intel Graphics and Visualization Institutes of XeLLENCE, the National Institute of General Medical Sciences of the National Institutes of Health under grant numbers P41 GM103545 and R24 GM136986 and the Department of Energy under grant number DE-FE0031880, and by the Deutsche Forschungsgemein-schaft (DFG, German Research Foundation) Project-ID 251654672 – TRR 161 (project B01). The GPU used for this research was donated by the Nvidia Corporation. The SPH dataset is provided by Stefan Reinhardt.

References

[1]. Bollobas B. Graph Theory: An Introductory Course. Springer-Verlag, New York, 1979. doi: 10.1007/978-1-4612-9967-7

[2]. Briais S, Caron S, Cioranesco J-M, Danger J-L, Guilley S, Jourdan J-H, Milchior A, Naccache D, and Porteboeuf T. 3D hardware canaries. In Prouff E and Schaumont P, eds., Cryptographic Hardware and Embedded Systems – CHES 2012, pp. 1–22. Springer, Berlin, Heidelberg, 2012.

[3]. Burton B, Aras K, Good W, Tate J, Zenger B, and MacLeod R. A framework for image-based modeling of acute myocardial ischemia using intramurally recorded extracellular potentials. Annals of Biomedical Engineering, 2018. doi: 10.1007/s10439-018-2048-0

[4]. Campbell PM, Devine KD, Flaherty JE, Gervasio LG, and Teresco JD. Dynamic octree load balancing using space-filling curves. Technical Report CS-03–01, Williams College Department of Computer Science, 2003.

[5]. Dafner R, Cohen-Or D, and Matias Y. Context-based space filling curves. Computer Graphics Forum, 19(3):209–218, 2000. doi: 10.1111/1467-8659.00413

[6]. Demir I, Dick C, and Westermann R. Multi-charts for comparative 3D ensemble visualization. IEEE Transactions on Visualization and Computer Graphics, 20(12):2694–2703, 2014. doi: 10.1109/TVCG.2014.2346448 [PubMed: 26356983]

[7]. Eichhammer E. Qt Plotting Widget QCustomPlot. https://www.qcustomplot.com/index.php/introduction, 2018.

[8]. Faloutsos C. Gray codes for partial match and range queries. IEEE Transactions on Software Engineering, 14(10):1381–1393, 1988. doi: 10.1109/32.6184

[9]. Genton M, Johnson C, Potter K, Stenchikov G, and Sun Y. Surface boxplots. Statistical Journal, 3(1):1–11, 2014.

[10]. Hilbert D. Ueber die stetige Abbildung einer Line auf ein Flächenstück. Mathematische Annalen, 38(3):459–460, 1891. doi: 10.1007/BF01199431

[11]. Itai A, Papadimitriou CH, and Szwarcfiter JL. Hamilton paths in grid graphs. SIAM Journal on Computing, 11(4):676–686, 1982. doi: 10.1137/0211056

[12]. Marcus DS, Wang TH, Parker J, Csernansky JG, Morris JC, and Buckner RL. Open access series of imaging studies (OASIS): Cross-sectional MRI data in young, middle aged, nondemented, and demented older adults. Journal of Cognitive Neuroscience, 19(9):1498–1507, 2007. doi: 10.1162/jocn.2007.19.9.1498 [PubMed: 17714011]

[13]. Matias Y and Shamir A. A video scrambling technique based on space filling curves (extended abstract). In Advances in Cryptology — CRYPTO ‘87, pp. 398–417, 1988. doi: 10.1007/3-540-48184-2 35

[14]. Mirzargar M, Whitaker RT, and Kirby RM. Curve boxplot: Generalization of boxplot for ensembles of curves. IEEE Transactions on Visualization and Computer Graphics, 20(12):2654–2663, 2014. doi: 10.1109/TVCG.2014.2346455 [PubMed: 26356979]

[15]. Mitchell WF. Hamiltonian paths through two- and three-dimensional grids. Journal of Research of the National Institute of Standards and Technology, 110(2):127–136, 2005. doi: 10.6028/jres.110.012 [PubMed: 27308109]

[16]. Netzel R and Weiskopf D. Hilbert attention maps for visualizing spatiotemporal gaze data. In Second Workshop on Eye Tracking and Visualization (ETVIS), pp. 21–25, 2016. doi: 10.1109/ETVIS.2016.7851160

[17]. Obermaier H, Bensema K, and Joy KI. Visual trends analysis in time-varying ensembles. IEEE Transactions on Visualization and Computer Graphics, 22(10):2331–2342, 2016. doi: 10.1109/TVCG.2015.2507592 [PubMed: 26685253]

[18]. Obermaier H and Joy KI. Future challenges for ensemble visualization. IEEE Computer Graphics and Applications, 34(3):8–11, 2014. doi: 10.1109/MCG.2014.52

[19]. Peano G. Sur une courbe, qui remplit toute une aire plane. Mathematische Annalen, 36(1):157–160, 1890. doi: 10.1007/BF01199438

[20]. Potter K, Wilson A, Bremer P-T, Williams D, Doutriaux C, Pascucci V, and Johnson C. Ensemble-Vis: A framework for the statistical visualization of ensemble data. In Proceedings of the 2009 IEEE International Conference on Data Mining Workshops, pp. 233–240, 2009.

[21]. Raj M, Mirzargar M, Preston JS, Kirby RM, and Whitaker RT. Evaluating shape alignment via ensemble visualization. IEEE Computer Graphics and Applications, 36(3):60–71, 2016. doi: 10.1109/MCG.2015.70 [PubMed: 26186768]

[22]. Reinhardt S, Huber M, Dumitrescu O, Krone M, Eberhardt B, and Weiskopf D. Visual Debugging of SPH Simulations. In 21st International Conference Information Visualisation (IV), pp. 117–126, 2017. doi: 10.1109/iV.2017.20

[23]. Rosen P, Burton B, Potter K, and Johnson C. muView: A visual analysis system for exploring uncertainty in myocardial ischemia simulations. In Visualization in Medicine and Life SciencesIII, pp. 49–69. Springer Nature, 2016. doi: 10.1007/978-3-319-24523-23

[24]. Sagan H. Space-Filling Curves. Springer-Verlag, New York, 1994. doi: 10.1007/978-1-4612-0871-6

[25]. Sedgewick R. Algorithms. Addison-Wesley Longman Publishing Co., Inc., Boston, USA, 1984.

[26]. Skubalska-Rafajłowicz E. Applications of the space-filling curves with data driven measure-preserving property. Nonlinear Analysis: Theory, Methods & Applications, 30(3):1305–1310, 1997. doi: 10.1016/S0362-546X(97)00277-0

[27]. Sun Y and Genton MG. Functional boxplots. Journal of Computational and Graphical Statistics, 20(2):316–334, 2011. doi: 10.1198/jcgs.2011.09224

[28]. Weissenböck J, Fröhler B, Gröller E, Kastner J, and Heinzl C. Dynamic volume lines: Visual comparison of 3D volumes through space-filling curves. IEEE Transactions on Visualization and Computer Graphics, 25(1):1040–1049, 2019. doi: 10.1109/TVCG.2018.2864510

[29]. Whitaker RT, Mirzargar M, and Kirby RM. Contour boxplots: A method for characterizing uncertainty in feature sets from simulation ensembles. IEEE Transactions on Visualization and Computer Graphics, 19(12):2713–2722, 2013. doi: 10.1109/TVCG.2013.143 [PubMed: 24051838]

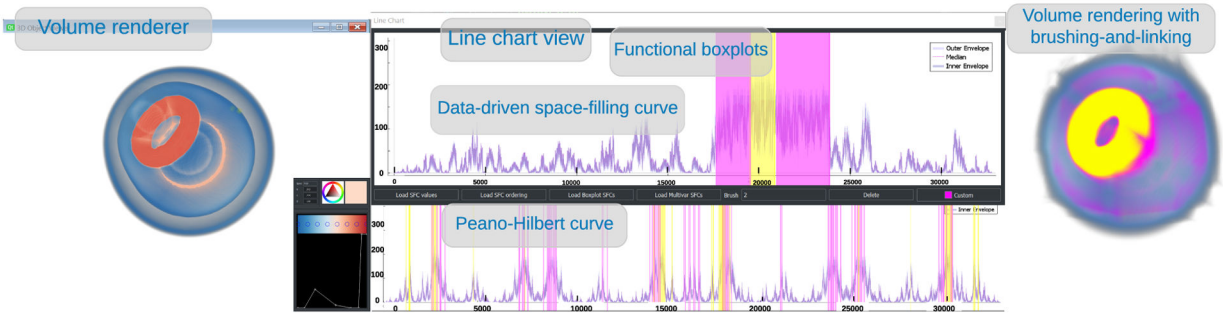

Fig. 1.

Visualizations of an ensemble of volumetric data. The key idea is to employ a mapping from 3D space (as in the volume renderings) to 1D space via space-filling curves, which then allow us to show boxplots of the data distributions. The ensemble is generated by sampling from Gaussian distributions of data values in the nucleon data with varying extents of uncertainty. The boxplots of the ensemble linearized with the Peano-Hilbert curve (bottom) do not preserve the coherency of 3D features—the small torus structure of high intensity cannot be readily identified. In contrast, our data-driven space-filling curve method (top) preserves features from 3D even in the 1D linearized representation as high intensities are more concentrated. This observation is confirmed by brushing-and-linking—the torus could be covered by one brush and its surroundings with two brushes with our method (see the volume rendering on the right and the yellow and purple regions in “Data-driven space-filling curve”), whereas multiple brushes are required by the Peano-Hilbert curve (yellow and purple regions in “Peano-Hilbert curve”).

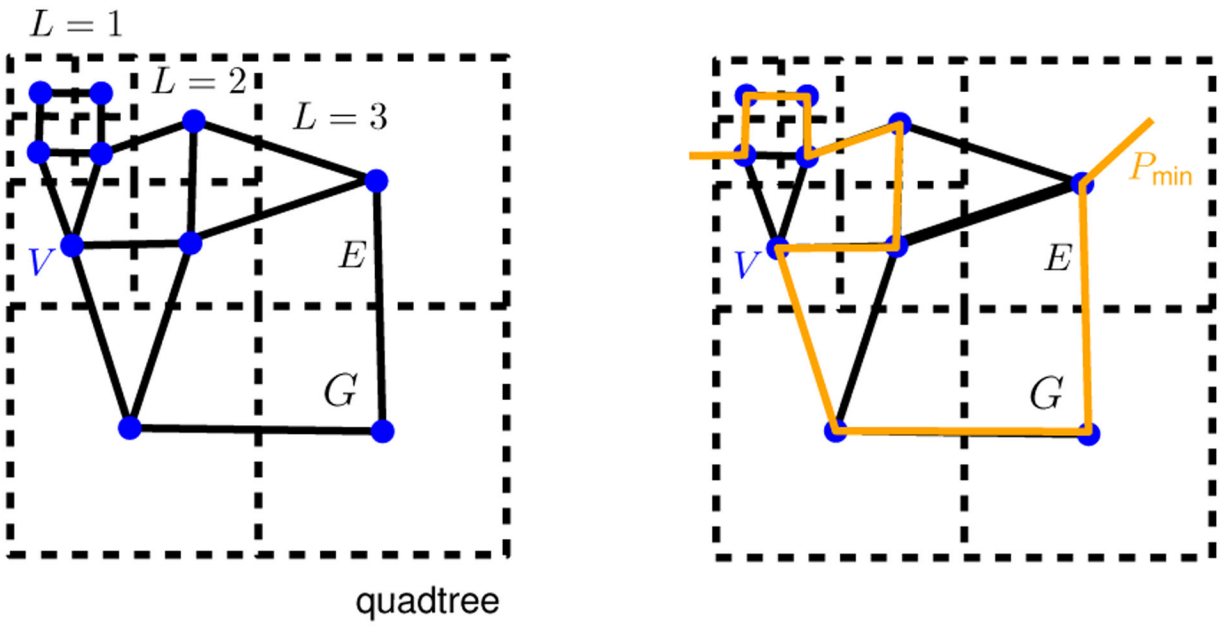

Fig. 2.
A 2D multiscale graph $G = (V, E, L)$ on a quadtree with a Hamiltonian path P_{min}.

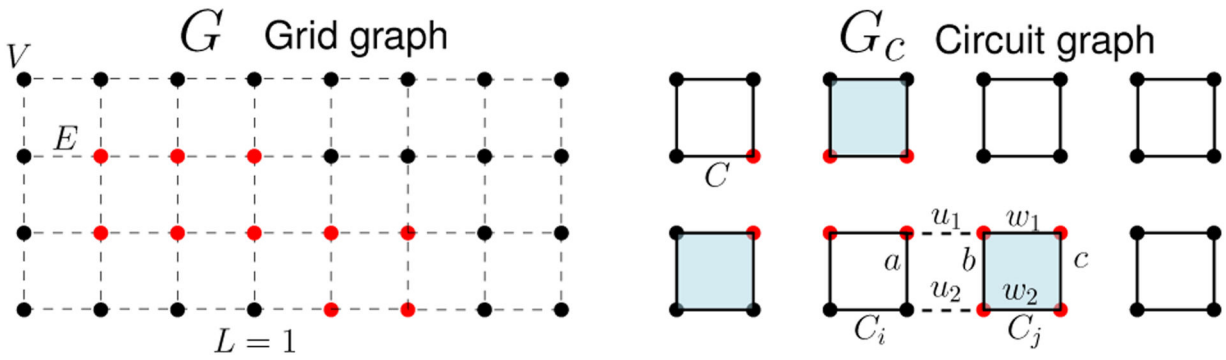

Fig. 3.

(Left) A 2D graph on a regular grid $G = (V, E, 1)$ with corresponding data values (black = 0, red = 1), and its associated circuit graph G_c (right) of circuits C. Adjacent circuits of C_i are drawn in light blue. The edge weights of data values between circuits C_i and C_j are shown on the right.

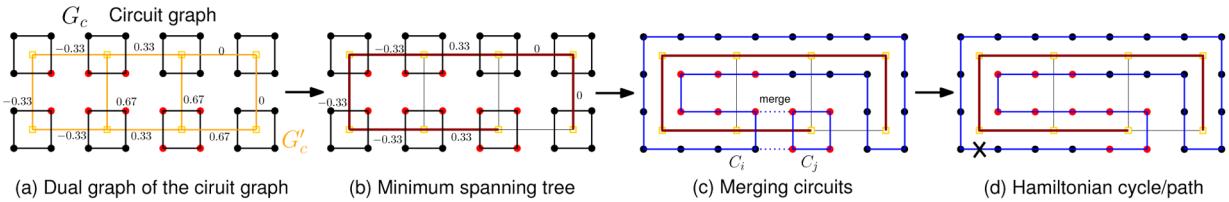

Fig. 4.

Intermediate steps of the framework of [5] on 2D regular grids. The dual graph G'_c. (a) of circuits graph G_c is a directed graph with our new weights (labeled on edges). Then, the minimum spanning tree (b) is found on the dual graph. Next, the Hamiltonian cycle is generated by merging circuits using the minimum spanning tree (c). Finally, (d) the Hamiltonian cycle is converted into a Hamiltonian path by making a cut anywhere on the cycle (shown as X).

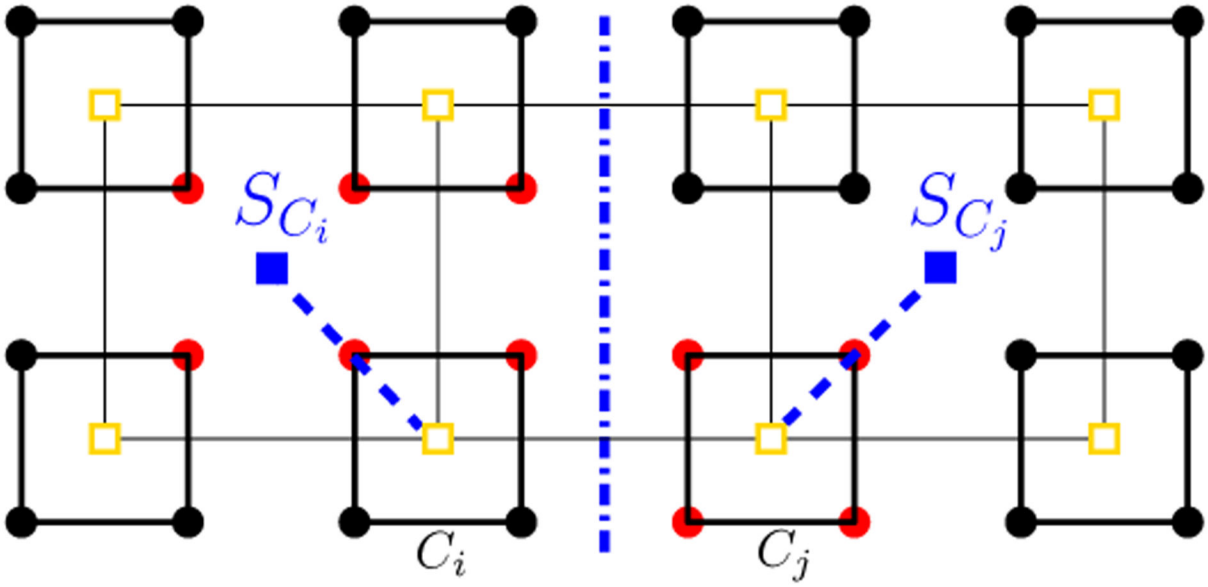

Fig. 5.

G_c' is partitioned (the blue dash-dot line) into blocks denoted by their centers (e.g., S_{C_i} and S_{C_j}) to accommodate the new positional term of the objective function.

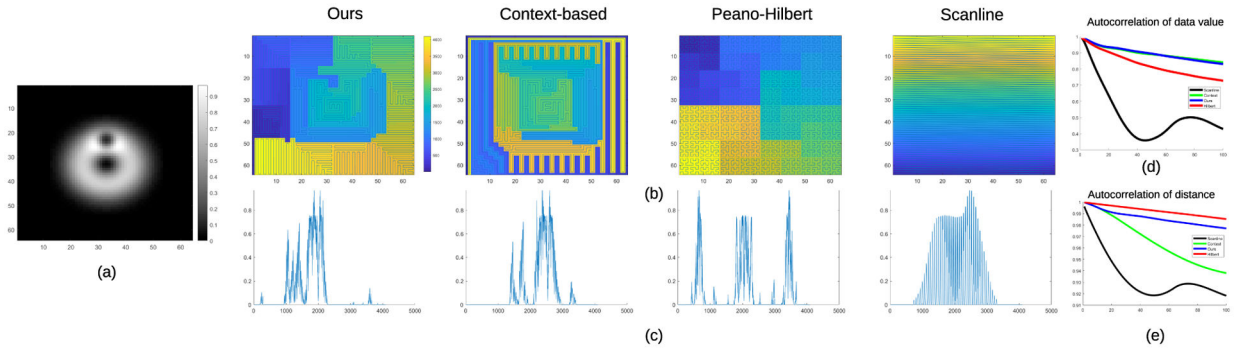

Fig. 6.

Comparison of linearization methods applied to a slice of the nucleon dataset (a). Our new data-driven space-filling curve is compared to previous methods: the context-based space-filling curve, the Peano-Hilbert curve, and scanline ordering. The spatial layout of the respective curves is shown in (b), and the linearization of the data values in (c). The spatial layout (b) is colored by the traversal order of curves (the horizontal axis of (c)) with the parula colormap (right of (b) Ours). Autocorrelations of value (d) and radial distance (e) quantify data coherency and locality preservation, respectively. The plots show that our approach provides the best compromise between the two conflicting goals. Note that the autocorrelations of data values of our method and the context-based method are largely overlapping.

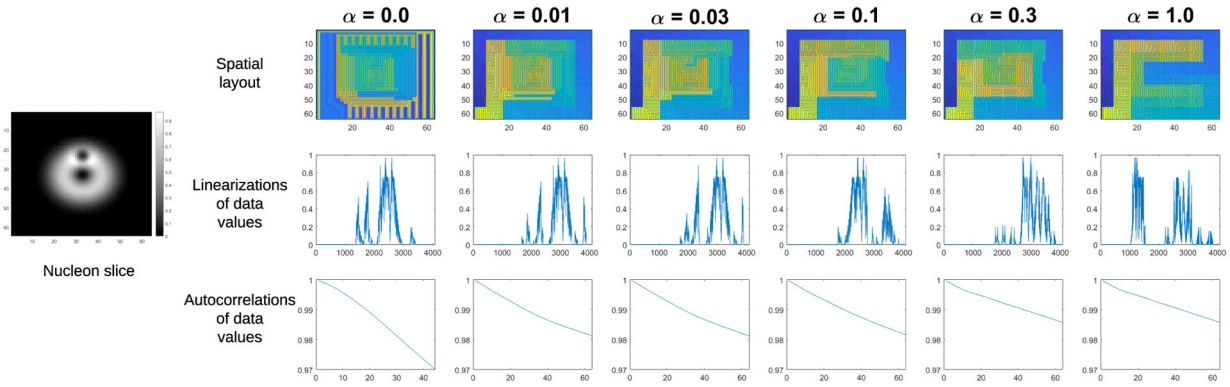

Fig. 7.

The effect of different α (left–right: 0, 0.01, 0.03, 0.1, 0.3, and 1.0) on the scan order of our space-filling curves for the nucleon slice data (left). The order is color-coded using the same color map as in Fig. 6.

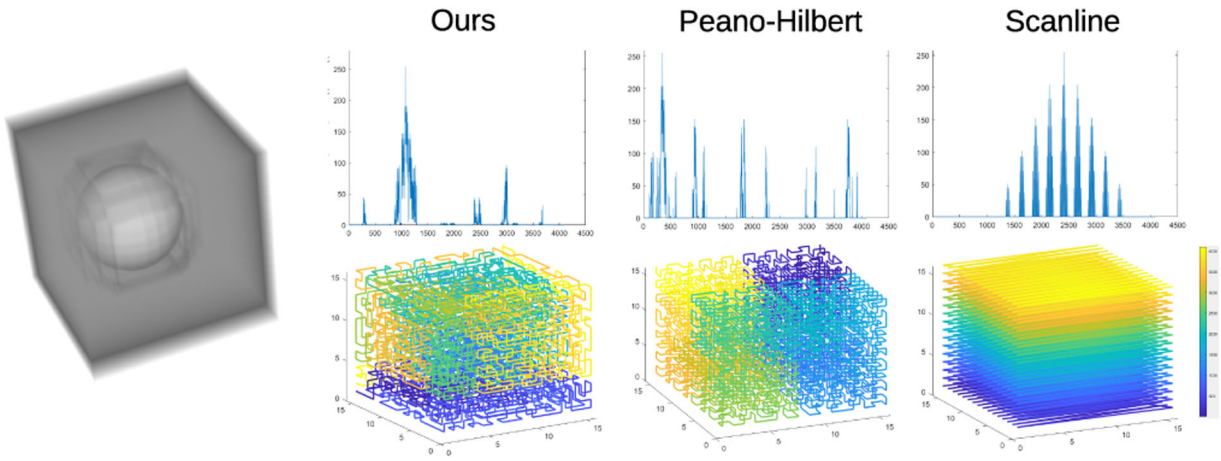

Fig. 8.

Linearizations (top) of a synthetic volume data of a sphere (left) with our data-driven curve, the Peano-Hilbert curve, and scanline ordering. The scan orders of curves are shown in the bottom row.

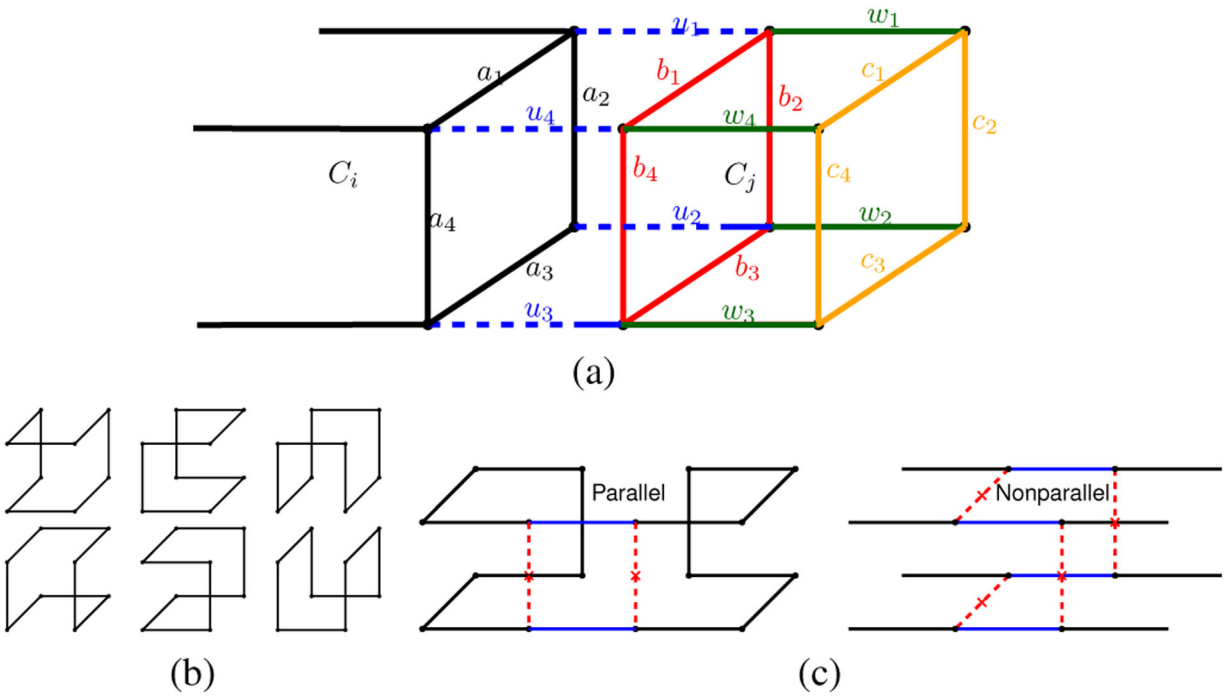

Fig. 9.

The value weights of cubes on 3D regular grids (a). The cubes need to be converted into cycles during merging. There exist (b) six cycle configurations of a unit cube, and the cycles are merged with (c) two association rules.

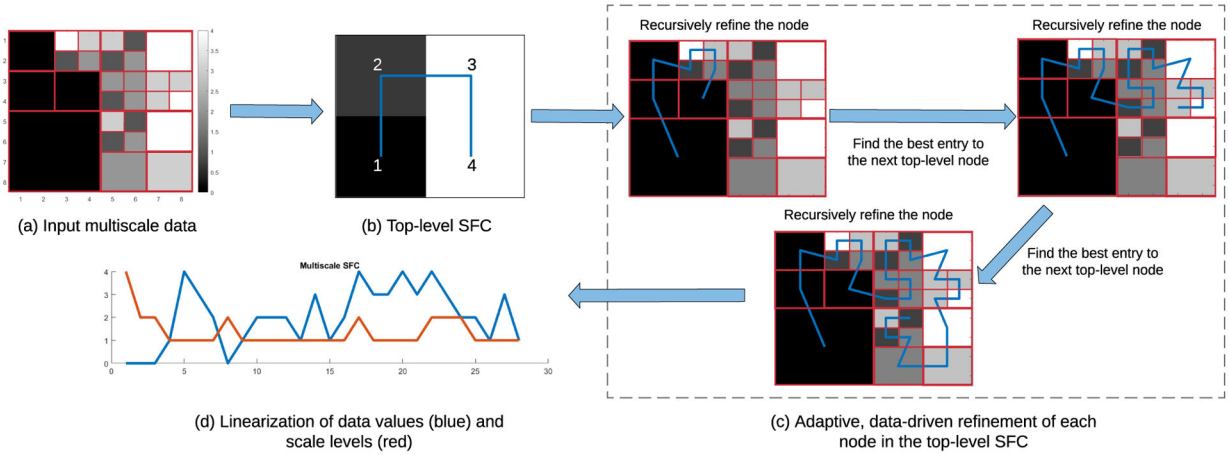

Fig. 10.

Steps to compute a data-driven space-filling curve for multiscale data.

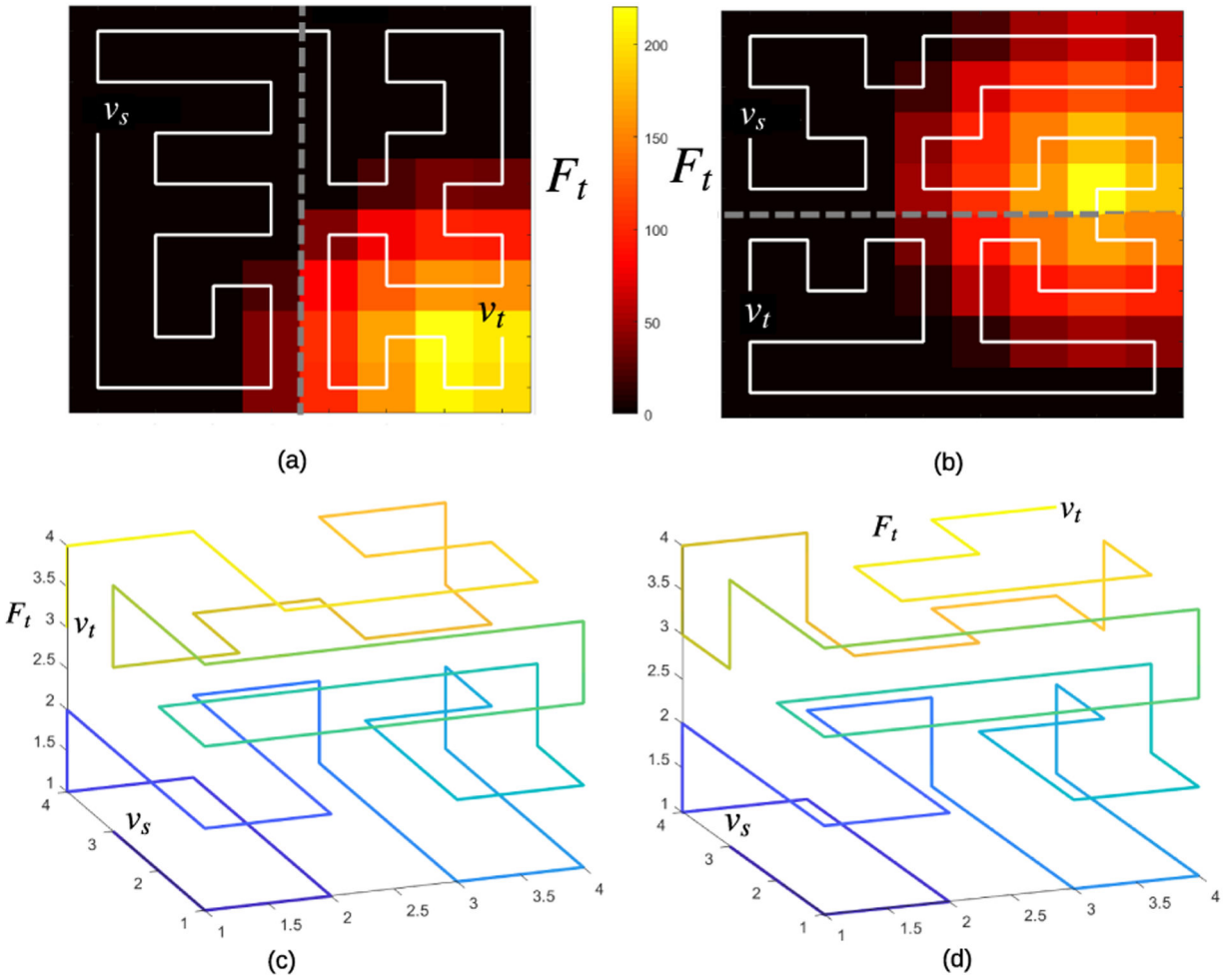

Fig. 11.

Flexible Hamiltonian paths for 2D and 3D grid graphs. Exit sides are on the (a) right and (b) left for these examples of 2D graphs. The example 3D graph has exit faces on the (c) left and at the (d) top.

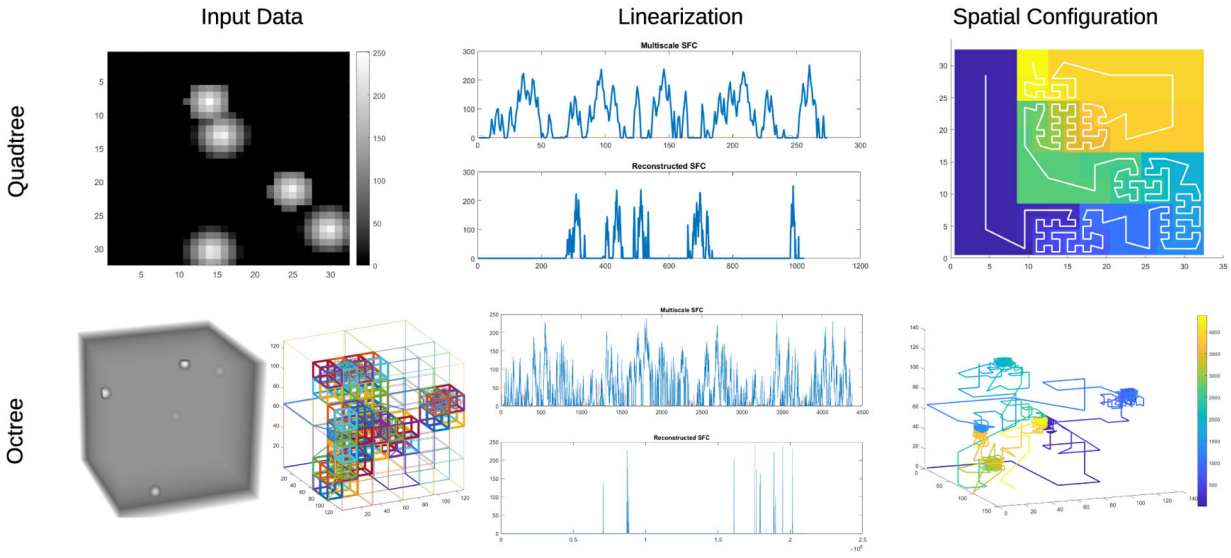

Fig. 12.

Data-driven space-filling curves for quadtree and octree. The input data are shown in the first column, the linearizations in the second column, and the spatial configurations of the space-filling curves in the third column.

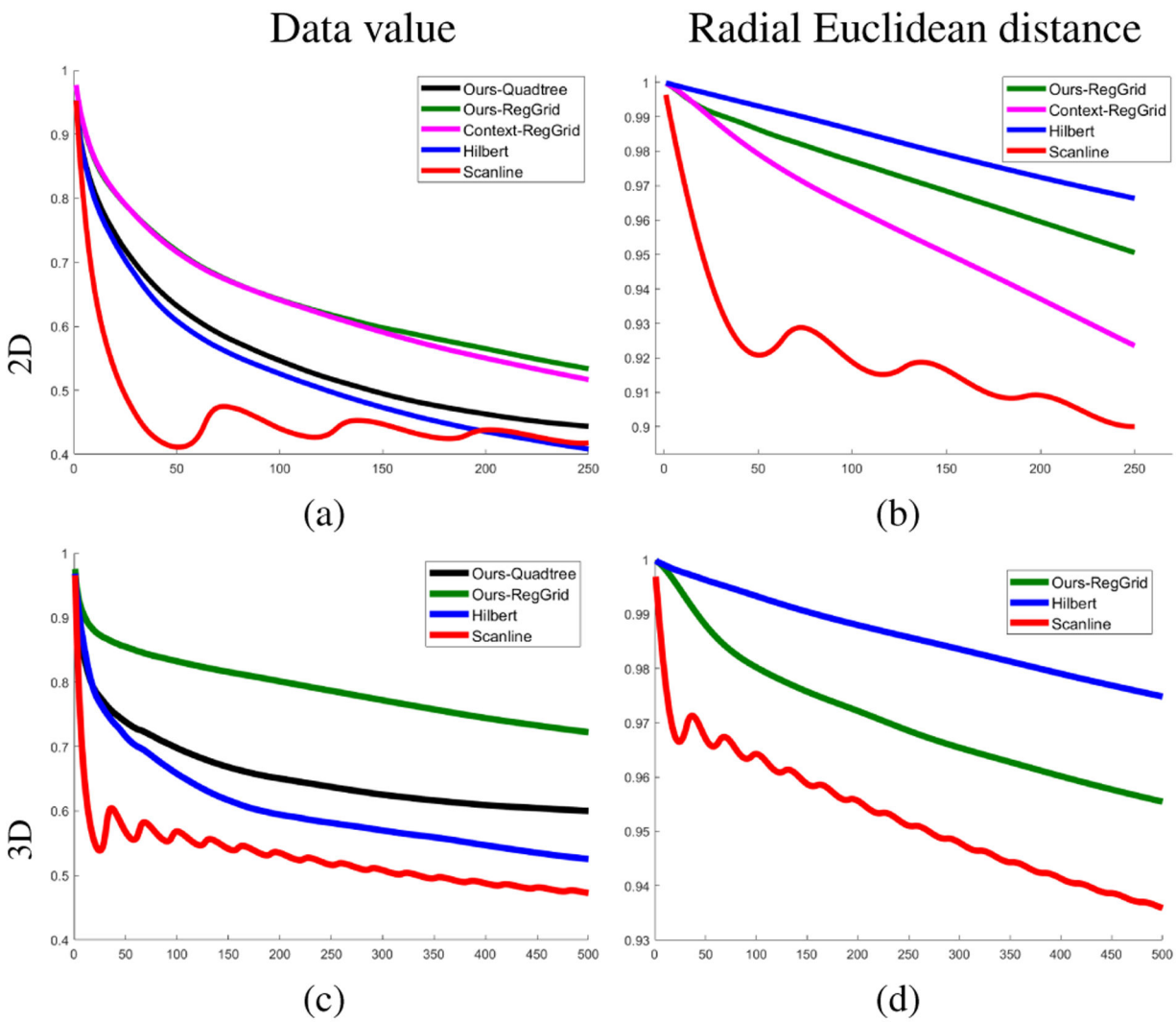

Fig. 13.

Autocorrelations of data value (first column) and radial distance (second column) for our 2D techniques (first row) and our 3D techniques (second row). Note that larger autocorrelation means better coherency.

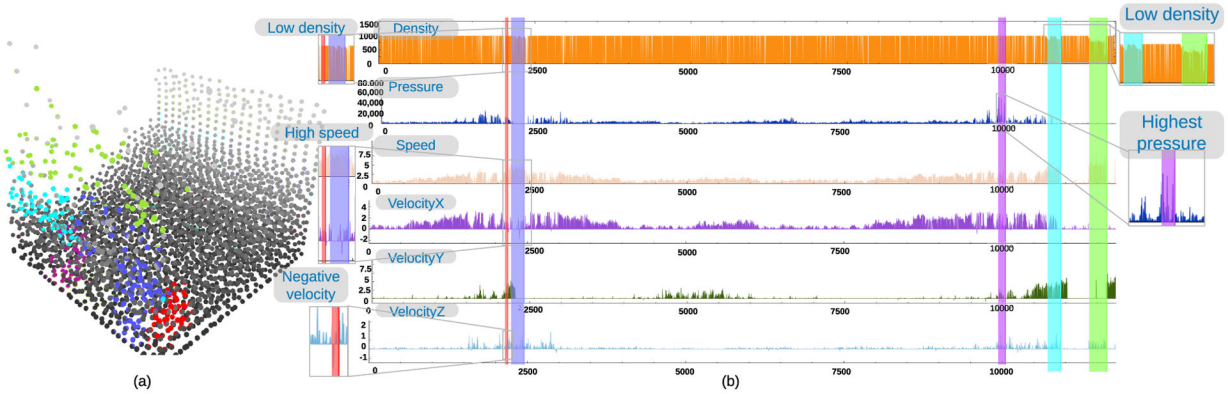

Fig. 14.

Visualization of an SPH simulation of dam break: (a) rendering of particles with brushed regions, (b) multivariate line charts generated using our octree-based data-driven space-filling curve.

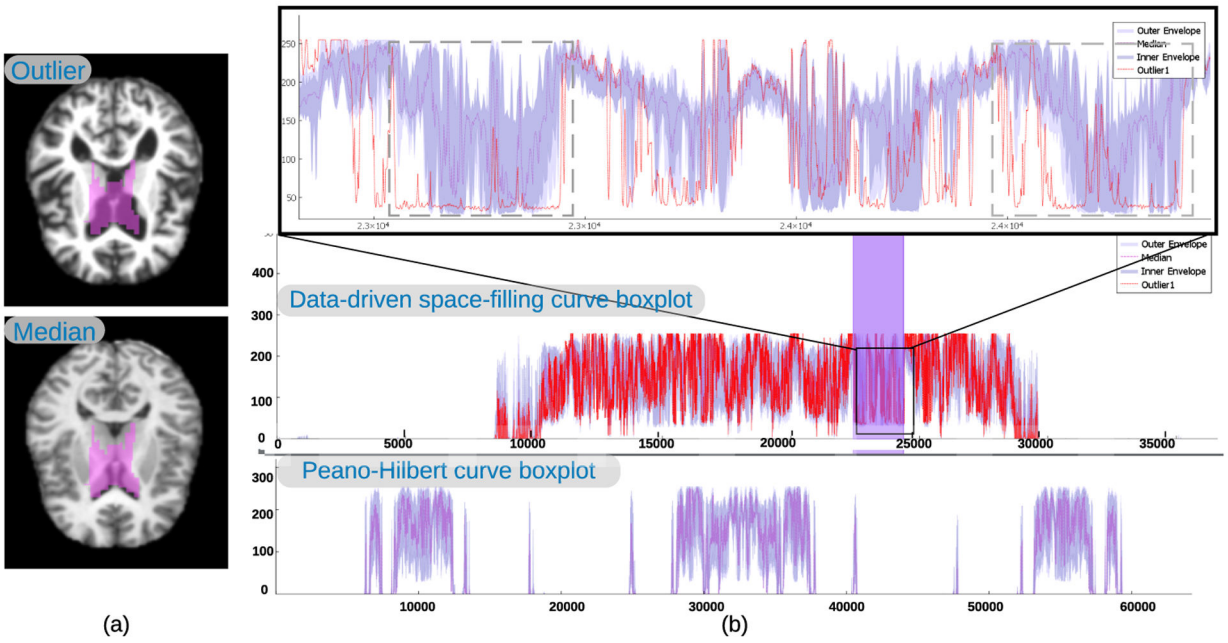

Fig. 15.

Visualization of a brain atlas with our data-driven space-filling curve and a Peano-Hilbert curve (with images padded to size of 256 × 256 with zeros). The median image (bottom) and an outlier image (top) are shown in (a) with brushing-and-linking (purple box) on the space-filling curves (b, center). The 1D layout with the space-filling curve allows for easy interaction, including brushing and zooming on spatial details, and it supports rendering boxplots that separate the brain from its surrounding and show that the outlier has wider low-value regions than the band at the lateral ventricle (red curves in the gray boxes in the zoom-in).

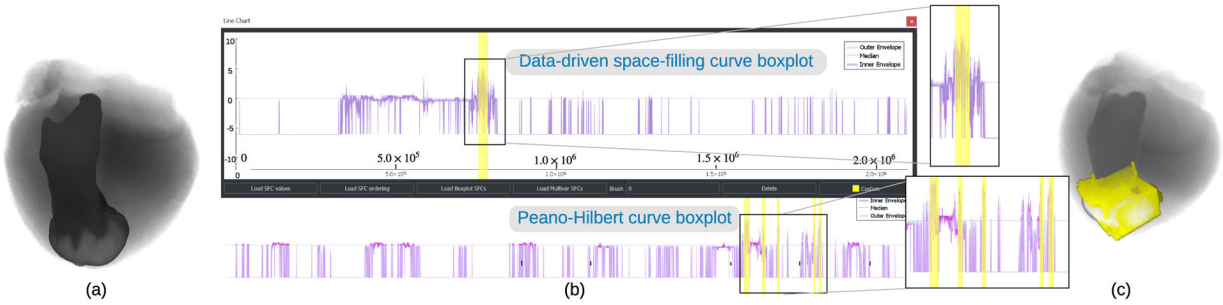

Fig. 16.

Ensemble visualizations of a heart ischemia simulation. The median member is volume-rendered in (a)—with 1D functional boxplots linearized with (b) our data-driven space-filling curve (top) and a Peano-Hilbert curve (bottom). The ischemic region that has potential value greater than 3 eV is selected in the boxplots and highlighted in (c).

Table 1.

Computation time of data driven space-filling curves on example datasets.

Dataset	Size	Time
Nucleon slice	64×64 pixels	12s
Nucleon	32×32×32 voxels	24s
Brain atlas	176×208 pixels	3m39s
SPH	4000 particles/11796 octree nodes	43s
Myocardial ischemia	128×128×128 voxels	4h31m