TransGraph: Hierarchical Exploration of Transition Relationships in Time-Varying Volumetric Data

Yi Gu and Chaoli Wang, *Member, IEEE*

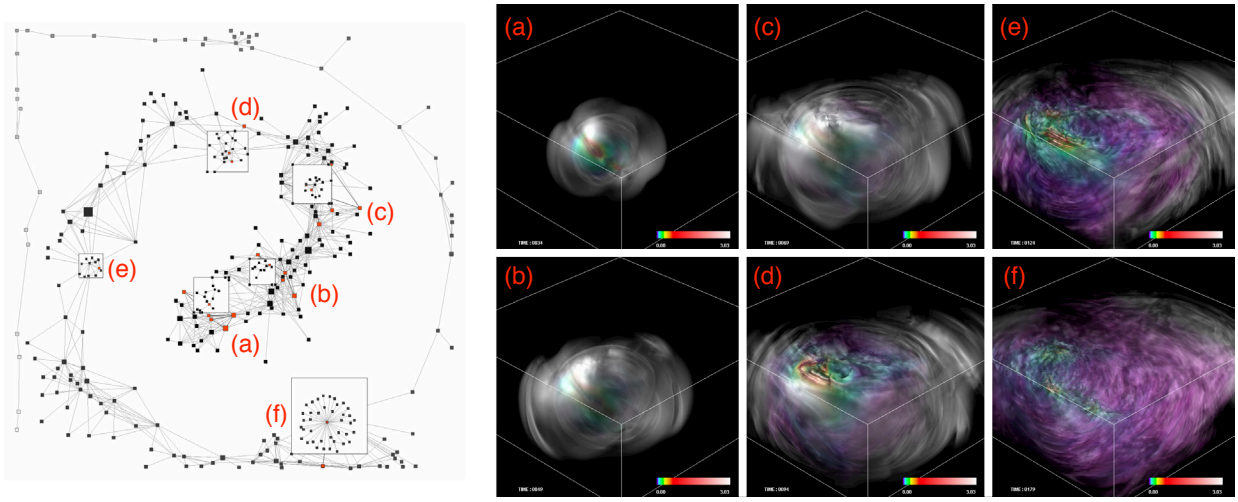


Fig. 1. Left: the TransGraph of the earthquake data set with dynamic tracking. We select a volume region corresponding to the earthquake's epicenter at time step 34, which is highlighted in its original color saturation in (a). Right: (a) to (f) are the dynamic tracking results in the volume at selected time steps 34, 49, 69, 94, 124, and 179, respectively. These images show the propagation of data transition over time. The corresponding nodes (states) and edges (transitions) at those time steps are highlighted in red and black, respectively, in the TransGraph.

Abstract—A fundamental challenge for time-varying volume data analysis and visualization is the lack of capability to observe and track data change or evolution in an occlusion-free, controllable, and adaptive fashion. In this paper, we propose to organize a time-varying data set into a hierarchy of states. By deriving transition probabilities among states, we construct a global map that captures the essential transition relationships in the time-varying data. We introduce the TransGraph, a graph-based representation to visualize hierarchical state transition relationships. The TransGraph not only provides a visual mapping that abstracts data evolution over time in different levels of detail, but also serves as a navigation tool that guides data exploration and tracking. The user interacts with the TransGraph and makes connection to the volumetric data through brushing and linking. A set of intuitive queries is provided to enable knowledge extraction from time-varying data. We test our approach with time-varying data sets of different characteristics and the results show that the TransGraph can effectively augment our ability in understanding time-varying data.

Index Terms—Time-varying data visualization, hierarchical representation, states, transition relationship, user interface.

1 Introduction

Time-varying volume data analysis and visualization is an important topic in scientific visualization. There exist several issues when analyzing and visualizing these four-dimensional space-time data. First, the spatial occlusion of data is inevitable due to the projection from 3D to 2D in the viewing. This prevents a simultaneous observation of the "entire" data especially when the time dimension is also considered. Second, when the data are visualized in an animated fashion, tracking data items or features over time could be very difficult as there might be different kinds of relationships present in the time series. Third, typical level-of-detail or hierarchical exploration of time-varying data only allows the user to navigate through the spatial and/or temporal domain in a coarse-to-fine fashion. The benefit of finding a good tradeoff between processing speed and visualization quality, however, may not actually help the user in terms of gaining more control in the analysis or leading to more insight from the data. To make it worse, all these issues are exacerbated by the ever-growing size and complexity of time-varying data we need to deal with.

The above issues call for solutions that extract data relationships over space and time, display these relationships in a low-dimensional space, and allow the user to perform queries and make connection to the spatiotemporal data. Explicit extracting data relationships makes it possible for the user to perform clear and quantified observation of data over time. Providing a user interface for navigation through the data and relationships essentially transforms the user from a *passive observer* to an *active participant* in the visual analysis process. Although there have been intensive research efforts spent on developing better algorithms and techniques for processing, managing, and rendering time-varying data, little attention has been paid to the design of effective user interfaces for supporting relationship exploration and detail examination. Transitions among data items or features over time capture the evolution of time-varying data and therefore are the most important relationships we seek to extract and visualize. In this paper,

• *The authors are with the Department of Computer Science, Michigan Technological University, 1400 Townsend Drive, Houghton, MI 49931. E-mail:{gyi, chaoliw}@mtu.edu.*

Manuscript received 31 March 2011; accepted 1 August 2011; posted online 23 October 2011; mailed on 14 October 2011.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

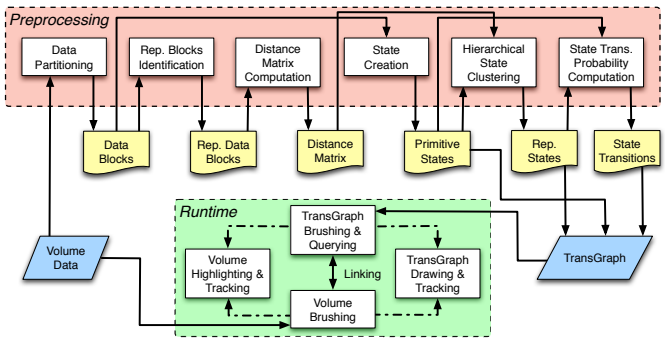


Fig. 2. The overview of our approach. We derive hierarchical state transition relationships from a given time-varying data set. Such relationships are visualized using a graph-based representation and integrated with volume view for visual data exploration.

we propose a novel graph-based user interface to enable systematic explorations of these transition relationships.

To allow cost-effective handling of large-scale time-varying data sets, we take a block-wise approach to analyzing the transition relationships. As shown in Figure 2, by constructing states from data blocks, clustering states into a hierarchy, and computing directional transition probabilities among states, we derive a hierarchical state transition graph, which we call the *TransGraph*, to capture the transition properties among data items over space and time. The TransGraph provides the user with an abstract, global mapping of the underlying time-varying data set as well as a user interface to assist interactive exploration of transition relationships. We employ a force-directed layout algorithm to draw the TransGraph, which enables the user to quickly identify important local regions by examining nodes and edges in the graph. It allows the user to pinpoint states of interest using a set of built-in queries and to track their evolution conveniently. Special care is taken to maintain the coherent update of the graph during level-of-detail exploration so that the user can observe relationships and track changes with ease. The user either brushes nodes in the TransGraph or regions in the original volume. These two views are linked together for cross referencing. Both views work hand-in-hand to enable knowledge extraction from the data, which is not possible with only a single view. We demonstrate the values of the TransGraph in assisting data exploration with experimental results gathered from several time-varying data sets of different characteristics.

2 Related Work

Utilizing the spatial and temporal coherence in time-varying data for efficient compression and rendering was the central focus of many early research efforts [23, 27, 11, 18, 25]. Recently, Fang et al. [9] proposed a simple yet effective idea of investigating time-varying data using time-activity curves (TACs), which record the values of voxels changing over time. This idea has been adopted by Wang et al. [26] to study the temporal importance of data blocks using information theory, by Woodring and Shen [29] to study the multiscale data clustering via wavelet transform, and by Lee and Shen [16] to study the temporal trend relationships with dynamic time warping. Time-varying data can be presented either statically or dynamically. Static representations include high-dimensional slicing and projection presented by Woodring et al. [30], video summarization presented by Daniel and Chen [7], and abstract storyboards presented by Lu and Shen [17]. Dynamic representations include storytelling [28, 17] and animation [2].

When the data are time-varying and multivariate or when many parameters are present for tweaking, mapping the data, variables, parameters, or features into a low-dimensional space can greatly facilitate the exploration and understanding of data relationships. Examples of this kind include the image graph presented by Ma [19] and the spreadsheet-like interface presented by Jankun-Kelly and Ma [14]. More recently, Mehta et al. [20] categorized three kinds of spatiotemporal relationships, i.e., directional, topological, and navigational relationships, among objects in time-varying scientific data and presented separate spatial and temporal graphs which are linked together through user interaction. Muigg et al. [21] proposed a four-level focus+context method to effectively reduce occlusion and cluttering among the huge number of function graphs and designed a highly-interactive function-similarity brush to allow temporal exploration of the data. Bruckner and Möller [5] presented a simulation system that allows the user to explore the simulation parameter space by generating samples, segmenting sequences, and clustering segments to assist visual effects design. For time-varying multivariate data, examples of effective visual representations and interfaces include the multifield-graphs presented by Sauber et al. [22] for multivariate correlation exploration, a tri-space interface presented by Akiba and Ma [1] for transfer function specification, and the attribute cloud presented by Jänicke et al. [12] for visual data analysis.

Closely related to our work is the work by Jänicke and Scheuermann [13] for flow feature analysis. Unlike their voxel-wise approach, we take a block-wise approach which is more amenable for efficient handling of large-scale time-varying data. Furthermore, in [13], causal states were defined based on the light-cones consisting the voxels at the current, past, and future time steps. The transitions between two states were defined based on the same spatial light-cone at time steps t and $t+1$. While we define transitions in a similar way, our definition of states is strictly restricted to data at the current time step. Therefore, in the resulting TransGraph we generate and visualize, there is a clear direction of time evolution. This is not presented in their ε-machine representation. As a result, we are able to track states and transitions *dynamically* both in the TransGraph and over the volume, while only *static* states and transitions can be tracked over the volume in [13]. Finally, we advocate an adaptive approach by building states from data blocks and clustering states into a hierarchy. By extracting a global view of data transitions, we are able to present states and transitions in an occlusion-free, controllable, and adaptive manner using graph-based visualization techniques.

3 Hierarchical State Transition

Figure 2 illustrates the input and output for each step of our approach to derive hierarchical state transition relationships. We first partition the volume data at each time step into blocks and identify representative blocks in a hierarchical manner. Representative blocks are used to construct a distance matrix to expedite the subsequent clustering. Next, we define states from data blocks and cluster states into a hierarchy. Finally, we derive directional transition probabilities among states to construct a hierarchical state transition graph. In the following, we describe each step in detail.

3.1 Data Blocks and Distance Measure

Given a time-varying volumetric data set, we perform uniform subdivision of each time step separately into a list of data blocks with an equal size. We will use the histograms of blocks to compute their distances, define states based on blocks, and cluster states hierarchically to extract their transition relationships. The size of block is chosen in a way so that each block is large enough for meaningful histogram computation. For instance, assuming a 256-level histogram, the block should contain at least a few thousands of voxels for stable computation. On the other hand, we should generate a sufficient number of blocks for effective hierarchical state clustering. At least a few hundreds of blocks for each time step is appropriate. In practice, we set the block sizes around 16^3 to 32^3 for the time-varying data sets we experiment with (see Table 1). For a data set with larger spatial and temporal dimensions, the block size should increase proportionally.

Given two data blocks b_i and b_j, we can compute their distance in many different ways. Instead of defining features and extracting them explicitly for comparison, we simply use the Jensen-Shannon divergence (JSD) to compute the distance between their probability distributions P_i and P_j. In probability theory and statistics, the JSD is a popular method of measuring the similarity between two probability distributions. As a symmetrized and smoothed version of the Kullback-Leibler divergence (KLD), the JSD has its minimum of 0.0

Algorithm 1 REPBLOCKSIDENTIFICATION(|w|: int; ε: float)

```
for each pass p do
    for each sliding time window w do
        if |w| = total # of time steps then
            b_w ← # blocks remaining in w
            b_max ← b_w
        else
            b_w ← # blocks remaining in w
            b_max ← # blocks in the first time step of w
        for b_i ← b_0 to b_max − 1 do
            if b_i has not been clustered then
                for b_j ← b_i to b_w − 1 do
                    if b_j has not been clustered then
                        if d_JS(b_i, b_j) ≤ ε then
                            Cluster b_i and b_j into the same group
    for each cluster c identified at pass p do
        Identify the rep. block as the one that has the smallest average distance
        to the rest of blocks in c
    Increase the size of sliding time window |w|
    Increase the distance threshold ε
    Gather the rep. blocks identified at pass p as the input for the next pass
Compute the distance matrix M for all rep. blocks identified in the last pass
```

Algorithm 2 HIERARCHICALSTATECLUSTERING(|w|: int; ε: float)

```
for each level l do
    for each sliding time window w do
        if |w| = total # of time steps then
            s_w ← # states remaining in w
            s_max ← s_w
        else
            s_w ← # states remaining in w
            s_max ← # states corresponding to the first time step of w
        for s_i ← s_0 to s_max − 1 do
            if s_i has not been clustered then
                for s_j ← s_i to s_w − 1 do
                    if s_j has not been clustered then
                        if s_i and s_j are spatially overlap or neighboring and are in
                        the same or neighboring time steps then
                            if p is the first pass then
                                d ← the average of the distances between their cor-
                                responding blocks in s_i and s_j looked up from M
                            else
                                d ← the distance between their primitive states in
                                s_i and s_j
                            if d ≤ ε then
                                Cluster s_i and s_j into the same group
    for each cluster c identified at level l do
        Identify the rep. state as the one that has the smallest average distance
        to the rest of states in c
    Increase the size of sliding time window |w|
    Increase the distance threshold ε
    Compute trans. probabilities p_{s_i→s_j} and p_{s_j→s_i} among all states at level l
    Gather the rep. states identified at level l as the input for the next level
```

when the two distributions are identical, and its maximum of 1.0 when they are least similar. All these properties nicely fit our requirement for distance measuring. Specifically, we define

$$d_{\mathrm{JS}}(P_i,P_j) = \big(d_{\mathrm{KL}}(P_i||\hat{P}) + d_{\mathrm{KL}}(P_j||\hat{P})\big)/2, \tag{1}$$

where $\hat{P}$ is the average of the two distributions, $\hat{P} = (P_i + P_j)/2$, and d_{KL} is the KLD, i.e.,

$$d_{\mathrm{KL}}(P_i||P_j) = \sum_{k=1}^{m} p_i(k) \log \frac{p_i(k)}{p_j(k)}. \tag{2}$$

In the discrete sense, we use a histogram to approximate the underlying probability distribution. Therefore, in Equation 2, m denotes the total number of bins in the histogram and $p_i(k)$ and $p_j(k)$ are the probabilities (normalized frequencies) for b_i and b_j corresponding to the kth bin. Given a data block, the histogram could be constructed simply as a one-dimensional histogram which records the distribution of the original scalar values, or as a multi-dimensional histogram which records the distribution of a combination of features. Since typical scientific data imply a continuous model, continuous histograms proposed by Bachthaler and Weiskopf [3, 4] can be used to build a better basis for the following computation and analysis. Unlike discrete histograms, continuous histograms are structurally independent of the resolution of a data set and could avoid misleading structures which are not part of the data but due to the low sampling resolution.

3.2 Representative Blocks and Distance Matrix

From partitioned blocks, we identify representative blocks and compute a distance matrix recording the distance between any two of them. Later on, for any two blocks in the volume, we will simply look up the distance between their representative blocks as the approximation instead of the actual distance computation. Since the number of representative blocks derived will be substantially less than the number of initial blocks acquired from volume partitioning, using such a small-size matrix for distance lookup will effectively reduce the cost of distance computation in the following hierarchical state classification. For a large input data set, computing the representatives from all blocks across all time steps would be very time consuming due to the large number of blocks considered simultaneously. We instead take a multi-pass process to efficiently identify representative blocks.

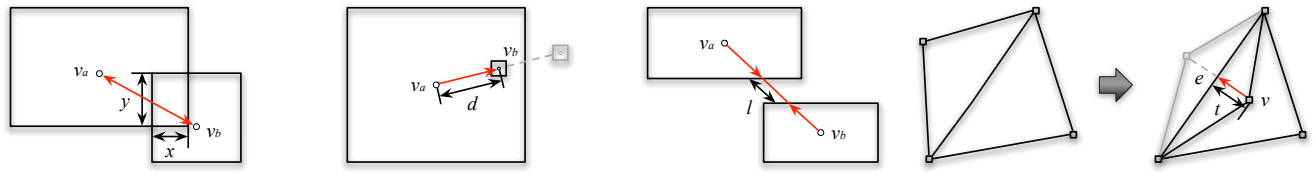
Our solution adopts a moving time window technique that considers a subset of data blocks at a time. Algorithm 1 summarizes our solution in pseudocode. In the first pass, the size of the time window is relatively small. We start b_i from the first block b_0 in the first time step of the initial window w_0 and cluster b_i with other blocks b_j in w_0 if their pairwise distance $d_{\mathrm{JS}}(b_i,b_j)$ (Equation 1) is less than a given distance threshold ε. Then, for the remaining blocks that have not been clustered, we locate the first block in the first time step of w_0 as b_i and perform the same clustering process again for these remaining blocks. This process for w_0 stops until we have attempted to cluster all its blocks where b_i loops through all the blocks in the first time step of w_0. Note that at this moment, it is likely that some blocks in w_0 might not be clustered. Next, we slide the window w_0 one time step further to create the window w_1 and perform the same clustering process for all blocks in w_1 that have not been previously clustered. This entire pass stops after the moving time window has swiped through the last time step. For each of the cluster we identify in the pass, we define its *representative block* as the one that has the smallest average distance to the rest of blocks in the cluster.

In the following passes, we repeat the same process as described above. The main differences are: (1) the size of time window will increase in each new pass, so does the value for the distance threshold ε; and (2) the input to the current pass is all representative blocks identified from the previous pass. In the last pass, the time window spans the entire time sequence. Only the representative blocks identified in this final pass will be used to build the distance matrix **M**. We point out that it could be very time-consuming to cluster blocks and identify representatives even though a multi-pass (essentially hierarchical) process is taken. In our work, we speed up the bottleneck computation (i.e., JSD calculation) using GPU with a CUDA implementation. The performance and speedup we achieve are reported in Section 5.1.

3.3 States and Transition Probabilities

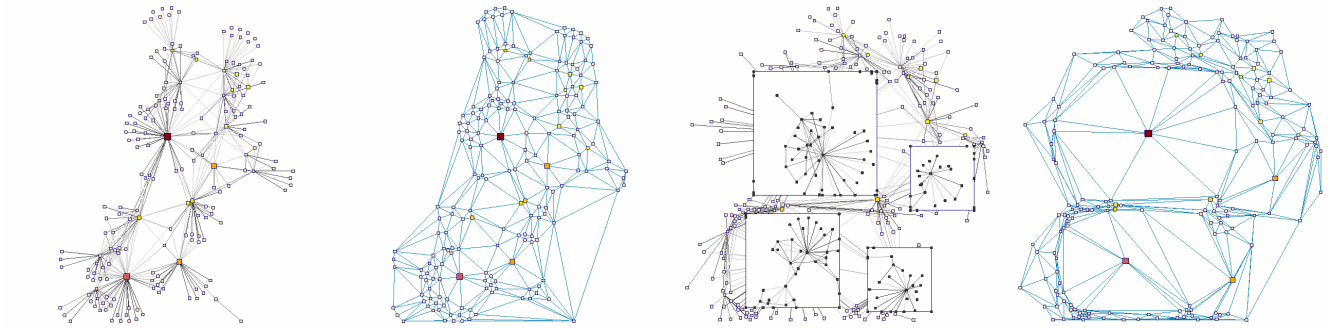
In this paper, we define *states* and their *transitions* as follows:

- A *state* is a configuration of neighboring spatiotemporal blocks characterized by their value distributions. A *primitive state* is a configuration of neighboring spatial blocks centered at each block at a single time step. In hierarchical state clustering, we select a *representative state* from a group of states. The group is formed by merging similar states that are spatially overlapping or neighboring and are in the same or neighboring time steps.



(a) bidirectional repulsive force (b) unidirectional repulsive force (c) spring force (d) attractive force

Fig. 3. The four different forces we consider for the TransGraph layout adjustment during the level-of-detail exploration. (a) the bidirectional repulsive force pulls apart two nodes that overlap each other. (b) the unidirectional repulsive force pushes a node away from an expanded node if it is inside the expanded node. (c) the spring force is introduced to keep the balance with respect to the two repulsive forces. (d) the attractive force handles the topology change of the underlying triangle mesh, i.e., when a triangle is flipped.



(a) initial layout (b) triangle mesh of (a) (c) final layout (d) triangle mesh of (c)

Fig. 4. (a) the initial layout at the coarsest level is produced using the Fruchterman-Reingold algorithm. The size of each node in the graph is proportional to the number of children within. (b) the triangle mesh produced from the initial node positions. (c) the layout after four nodes are selected and expanded for detail examination. Four different forces (Figure 3) are used in the adjustment. (d) the underlying triangle mesh is used to maintain the topology of the graph during layout adjustment.

- A *transition* occurs between two states, from one state at time step t to another state at time step $t+1$, if and only if their corresponding central blocks are the same in the spatial domain. Given two states s_i and s_j at the same clustering level, the directional *transition probability* $p_{s_i \to s_j}$ is the ratio between the total number of transitions from s_i to s_j and the total number of transitions from s_i to all states (including itself). This definition indicates how frequently a state is transferred to another state.

To create primitive states, we use an arrangement of $3 \times 3 \times 3$ blocks for spatial configuration. The number of primitive states is the same as the number of data blocks. As shown in Algorithm 2, for all primitive states in the data set, we perform hierarchical clustering and select a representative state from each cluster. We use a method similar to region growing in our clustering mainly due to the efficiency concern. The spatial and temporal constraints are to ensure that clustered states are space-time continuous which makes the subsequent tracking process meaningful. For distance computation between two primitive states s_i and s_j, if any of them is along the volume boundary, special care is taken so that their partial block correspondence is used to derive the distance. Unlike state clustering, we only consider the central blocks in primitive states for transition probability computation. If both states s_i and s_j are clusters of states rather than primitive states, then the transition probability is computed based on the total number of transitions from any primitive state in s_i to any primitive state in s_j. As needed, we may consider the transition probability for all states either in each individual time step (local normalization) or across all time steps (global normalization). In the first case, it would lead to a time-varying graph highlighting the dynamics transitional relationship changes over time. In the second case, it would lead to a static graph summarizing all transitions across all time steps.

4 TransGraph

We create a hierarchical state transition graph, i.e., TransGraph, to record the transition relationships among states at various levels of detail. In this way, we convert a 4D space-time volume data set to a graph which can be visualized in 2D. In the TransGraph, a node represents a state (leaf) or a cluster of states (non-leaf) and the edge between two nodes represents their transition probabilities. For simplicity, we draw a single undirected edge instead of double directed edges in the TransGraph. The edge weight is the summation of the two directional transition probabilities associated with the two incident nodes. Note that the summed edge weight is used only for the purpose of graph drawing while the underlying graph representation is still a directed graph. This allows the user to perform separate transition queries on incoming or outgoing transition probabilities. In the following, we describe how we draw the TransGraph to account for the level-of-detail adjustment, and how we brush, query, and track in the graph view in conjunction with the volume view to enable visual data analysis and understanding.

4.1 TransGraph Drawing

The TransGraph has two places that introduce dynamics into graph drawing. First, the nodes and edges in the TransGraph change for every time step. Second, the level-of-detail exploration leads to graph changes during the interaction, which requires adjusting node positions to reduce overlap or occlusion. A good layout for TransGraph drawing should maintain a good balance between preserving the mental map (i.e., the abstract structural information a user forms by looking at the graph layout) and revealing the dynamics.

In this paper, we adopt the idea of the supergraph [8] by computing a global layout which induces a layout for each time step. This allows us to observe all states and their transitions in a single visualization. In the meanwhile, each graph in the time sequence becomes a subset of the supergraph and is visualized sequentially. To support effective graph viewing, we propose a constrained layout adjustment algorithm to maintain coherent update and minimize node overlap during interactive level-of-detail exploration.

We employ the Fruchterman-Reingold algorithm [10], a popular force-directed layout algorithm to all levels of TransGraph drawing. This algorithm ensures that topologically near nodes are placed near to each other and topologically far nodes are placed far from each other.

Table 1. The timing results and parameters used in histogram and JSD computation. The timing (in seconds) for data read, histogram and JSD computation is the total time for all time steps.

data set	variable	volume dimension	block size	data read	hist comp	# bins	JSD (CPU)	JSD (GPU)	two-pass time win	two-pass threshold ε
climate	salinity	$360 \times 66 \times 27 \times 120$	$15 \times 11 \times 9$	0.23	0.60	64	58	19.2	3, 120	0.05, 0.1
			$15 \times 6 \times 9$	0.26	0.73	64	155	7.2	6, 120	0.1, 0.15
			$30 \times 11 \times 9$	0.24	0.99	128	41	4.8	5, 120	0.1, 0.15
combustion	YOH	$480 \times 720 \times 120 \times 122$	$30 \times 30 \times 30$	167	78	256	12384	31	4, 122	0.05, 0.1
earthquake	amplitude	$256 \times 256 \times 96 \times 599$	$16 \times 16 \times 16$	125	53	256	64883	95	3, 599	0.25, 0.3
			$16 \times 16 \times 8$	119	23	128	154212	233	4, 599	0.075, 0.085
			$32 \times 32 \times 16$	119	41	512	3982	17.9	4, 599	0.075, 0.09
hurricane	vapor	$500 \times 500 \times 100 \times 48$	$20 \times 20 \times 20$	34	18.2	256	23058	16.3	4, 48	0.05, 0.15
ionization	gas temp.	$600 \times 248 \times 248 \times 200$	$30 \times 31 \times 31$	164	114	256	73156	156	4, 200	0.0135, 0.02

Table 2. Continuation of Table 1 with the first column re-listed here. The timing results and parameters used for hierarchical clustering. The timing (in seconds) for clustering is the total time for all time steps.

data set	clustering (CPU)	three-level time win	three-level threshold ε
climate	52	7, 120, 120	0.15, 0.3, 0.45
	118	6, 120, 120	0.275, 0.425, 0.575
	7.6	3, 120, 120	0.275, 0.375, 0.45
combustion	628	5, 122, 122	0.2, 0.35, 0.4
earthquake	1060	5, 599, 599	0.2, 0.4, 0.6
	17059	2, 599, 599	0.2, 0.3, 0.4
	222	3, 599, 599	0.2, 0.25, 0.4
hurricane	5069	7, 48, 48	0.175, 0.3, 0.45
ionization	1027	5, 200, 200	0.2, 0.45, 0.7

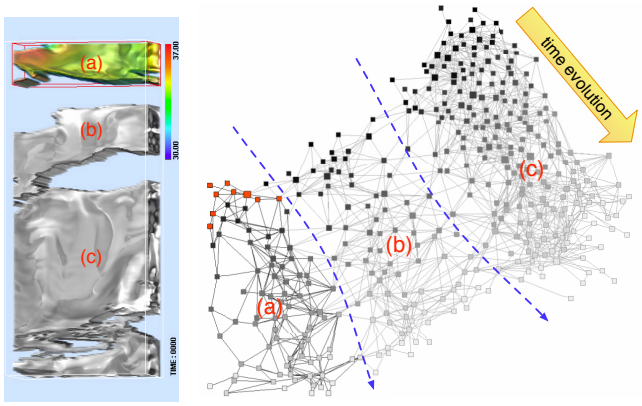


Fig. 5. Top: a spatial region at the first time step of the climate data set is selected and highlighted in its original color saturation. Bottom: the corresponding nodes are highlighted in red in the TransGraph with the subsequent transitions (edges) of this region highlighted in black. The rest of nodes are displayed with decreasing shading as the time step increases to indicate the direction of time evolution in the graph. Three well-isolated spatial regions marked with (a), (b), and (c) form three clear-cut clusters in the TransGraph.

Furthermore, unlike other algorithms such as the Kamada-Kawai algorithm [15], nodes will not get too close to each other in the drawing. Therefore, the drawing area is effectively utilized as overlap or occlusion among nodes is reduced.

During the graph exploration, the user selects one or multiple nodes for examining their higher levels of detail. As such, selected nodes would be expanded which demands layout adjustment. We generate the initial layout for the coarsest level of the TransGraph. To preserve the relative positions of nodes in the initial layout for coherent update, we apply the triangulation scheme proposed by Shewchuk [24] to the initial graph and use the result of the triangulation to perform constrained layout adjustment when nodes are expanded for further examination. When such a node is expanded, its initial size is proportional to the number of children in its next level of detail. All nodes expanded are assigned the same scaling factor. Similar to the work of dynamic word cloud visualization by Cui et al. [6], we consider the following forces to reposition the nodes to reduce their overlap while maintaining the topology of the coarsest level of the TransGraph.

- **Bidirectional repulsive force:** This force pushes away two nodes v_a and v_b from each other and is effective if and only if v_a and v_b overlap each other. The bidirectional repulsive force is defined as $f_1(v_a, v_b) = k_1 \times \min(x, y)$, where k_1 is a given weight and x and y are the width and height of the overlapping region as shown in Figure 3 (a). This force is applied to every pair of nodes in the TransGraph.

- **Unidirectional repulsive force:** This force pushes away a node v_b without detail shown from a node v_a with detail shown and is effective if and only if v_b is inside v_a. The unidirectional repulsive force is defined as $f_2(v_a, v_b) = k_2/d$, where k_2 is a given weight and d is the distance between the centers of v_a and v_b, as shown in Figure 3 (b). If d is close to zero, then an upper-bound force $f_{2\max}$ is used instead which guarantees that v_b will be moved outside of v_a.

- **Spring force:** This force is used to balance the graph by offsetting the two repulsive forces introduced. Given two nodes v_a and v_b, the spring force is defined as $f_3(v_a, v_b) = w_a \times w_b \times l$, where w_a and w_b are the weights of v_a and v_b respectively, and l is the length of the edge connecting the centers of v_a and v_b that lies outside of their boundaries as shown in Figure 3 (c). We compute w_a and w_b based on the number of children in v_a and v_b, respectively. The larger the number, the higher the weight. This force is applied to every pair of nodes in the TransGraph.

- **Attractive force:** This force is used to maintain the underlying triangle mesh we construct for the coarsest level of the TransGraph. During the layout adjustment, if a mesh triangle is flipped, as shown in Figure 3 (d), then the topology of the triangle mesh changes. Our goal is to maintain a stable update of the graph by introducing an attractive force to flip the triangle back. The attractive force is define as $f_4(v_a) = k_4 \times t$, where k_4 is a given weight and t is the distance from node v to edge e. We also consider virtual triangle edges connecting extreme nodes in the graph to the four corners of the drawing area. This is to ensure that all graph nodes do not go out of bound.

Figure 4 shows an example of layout adjustment during the level-of-detail exploration. As we can see, the expanded nodes expel other nodes outside of their regions while the global structure of the TransGraph is still preserved. In this example, we give a fairly large size for drawing each of the higher levels-of-detail to show an extreme case. In

practice, the size for drawing a higher level-of-detail could be smaller as long as the nodes within can be clearly seen without much clutter, as shown in Figure 6. Our layout adjustment strategy is applied recursively to different hierarchical levels in the same manner.

4.2 TransGraph Query

The TransGraph stores rich information about the nodes (states) and edges (transitions) over time. To make the best use of the TransGraph, we propose a set of intuitive queries to enable knowledge extraction from the underlying time-varying data.

- **State query:** We derive how active a state is by computing the number of time steps the state remains active. By summarizing its connecting states and their probabilities, we can tell how active a node is in terms of transferring to other states or how inactive it is in terms of staying in the same state. From the hierarchy of the TransGraph, we also derive the active volume extent a state corresponds to by summarizing the information of its descendants. These queries would allow the user to automatically locate important or dominant states with respect to the spatial extent, transition relationship, or time span. Such queries shares some similarity with recognizing node centrality in a graph, while our query provides quantified search that is unavailable by simply observing the TransGraph.

- **Transition query:** From the edges of the TransGraph, we derive which transitions are the most dominant ones over time and what are the corresponding states. We can tell which transitions are the most or least balanced ones by comparing the two directional transition probabilities associated with each edge. From the hierarchy of the TransGraph, we also perform transition queries recursively for finer-grained examination as needed.

- **Time step query:** We identify important time steps in the time sequence by summarizing and ranking time steps based on the number of states active, the number of transitions active, or the number of states involved in self- or non-self transition only. Such temporal queries can be used to automatically highlight important or interesting time steps where state transitions are much more frequent or widespread than other time steps.

4.3 Brushing and Tracking

We dynamically link together the two views of the time-varying data, namely, the volume view and the graph view. The user interacts with the data in one view and the result is automatically reflected in the other view. This dual-domain interaction is quite standard for the visualization of data from different perspectives. The brushing and linking between the two views compliments each other. The volume view is intuitive for understanding while data occlusion is inevitable and data selection could be tricky. On the contrary, the graph view is an abstract representation while occlusion-free data selection is fairly straightforward. Combining both views together allows easy interaction and comprehensive understanding of the time-varying data.

The user selects data blocks either directly from the volume (by bounding the ranges in the x, y, and z directions) or from the TransGraph (by direct clicking or range selecting the states of interest). For block selection from the volume, the user can clip the volume or adjust the transfer function to better identify interesting 3D data blocks that are occluded. We allow tracking of these data blocks over the space and time by highlighting the influenced region in the volume and the corresponding states in the TransGraph.

We present two different ways of tracking: *static tracking* and *dynamic tracking*. Static tracking is to fix the spatial blocks while tracking their state changes over time. Tracking over the TransGraph only is straightforward as all transition information is recorded beforehand. Unlike static tracking, dynamic tracking also conveys the impression of data transition in the volume while tracking the corresponding state changes over time. To enable volumetric tracking, we modulate the color saturation of data blocks based on their transition probabilities. For simplicity, we assume a constant propagation speed of the data. The user has the flexibility to adjust the speed at runtime. Adding the propagation speed enables us to fine tune data transition at the voxel-wise level, going beyond the block-wise granularity for state transition computation. In addition, we consider the distances of a voxel to the centers of initial blocks selected for tracking to further adjust the color saturation of the voxel (refer to Section 5.4 for more detail). This allows us to differentiate voxels within a block with different color saturations even though the transition probability stays in the block-wise level. In practice, we perform all these color modulations in the fragment program to ensure the interactivity and realtime tracking.

5 Results and Discussion

In this section, we first report the data sets we experimented with and their timing performance. Then, we use different data sets to demonstrate how brushing and linking, query and tracking are performed between the volume view and the graph view. We also describe what are the knowledge and benefit gained from such a dual-domain interaction. In addition, we show the TransGraph layouts with different parameter settings to study their influence on the graph layout.

5.1 Data Sets and Timing Performance

We experimented our approach with five time-varying volumetric data sets. All these data sets are produced from scientific simulations. From top to bottom in Table 1, the data sets are from a simulation of the equatorial upper-ocean, a simulation of turbulent combustion, a simulation of Northridge earthquake in 1994, a simulation of Hurricane Isabel in 2003, and a simulation of ionization front instabilities. We picked one variable from each data set in our study.

The timing was collected on a PC with an Intel 2.4GHz CPU and an nVidia GeForce GTX 465 GPU. There are three major tasks in our computation: histogram computation, JSD computation, and hierarchical clustering. Tables 1 and 2 show the timing results. Histogram computation is to calculate the histograms for all blocks in the data. JSD computation is to compute the distance among blocks to identify representative blocks, and to compute the distance matrix for distance lookup in the subsequent clustering. Hierarchical clustering is to cluster states into a hierarchy for building the TransGraph. For JSD computation, we opted for two passes for simplicity. For hierarchical clustering, we opted for three levels (not including the leaf level consisting of all primitive states). After the initial clustering, we did not impose the time window anymore so that more neighboring states can be merged together into clusters. Among the three tasks, the bottleneck is JSD computation because of the large numbers of log operations (Equation 2) involved. As such, we also implemented a GPU version to speed it up by calculating all pairs of JSDs for a group of blocks in parallel. All other tasks were computed in the CPU.

For the TransGraph drawing, the initial graph layout and triangulation only needs to be computed once for each data set, which can be completed within a few seconds. At runtime, we allow the user to interactively explore the TransGraph at various levels of detail and perform layout adjustment in real time with up to thousands of nodes. Beyond that, the performance may not be interactive and the drawing area may not be sufficient. Therefore, in our current implementation, we do not allow the user to explore too deep down the TransGraph, such as the leaf level.

5.2 Brushing and Linking

The brushing and linking between the volume view and the graph view provides the user with an intuitive way to understand both data and make connections. For example, Figure 5 shows an example where we brush a spatial region corresponding to the equatorial Indian Ocean of the climate data set. In the TransGraph, the edges (transitions) associated across all time steps are highlighted in black and the nodes (states) corresponding to the first time step are highlighted in red. The rest of nodes are displayed with decreasing shading as the time step increases. In this way, the direction of time evolution is easily discernible from the graph. As we can see, the well-isolated spatial region we brush forms a clear-cut cluster in the TransGraph. As a matter of fact, three

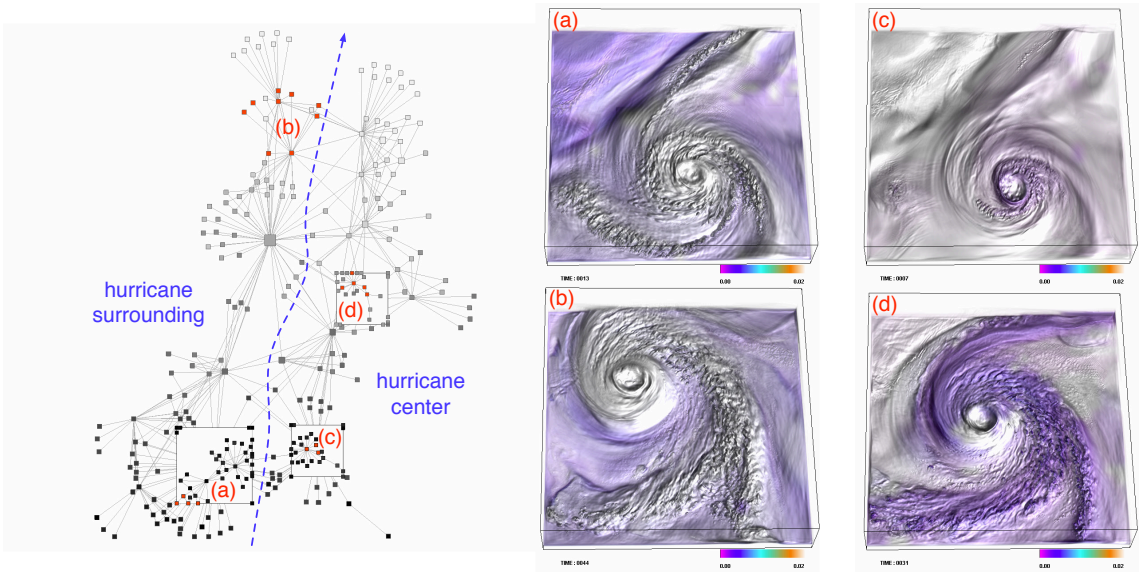


Fig. 6. Left: the TransGraph of the hurricane data set. Right: (a) to (d) are the corresponding volume highlighting indicating the red nodes selected in the TransGraph at time steps 13, 44, 7, and 31, respectively. The spatiotemporal regions corresponding to the hurricane's surrounding, i.e., (a) and (b), and center, i.e., (c) and (d), lie on the two clear-cut parts of the TransGraph, respectively.

distinct node clusters are fairly well separated in the TransGraph which correspond to the three ocean regions separated by the continents (i.e., void regions in the volume view). Thanks to the hierarchical state clustering which imposes that the clustered states must be spatiotemporal neighbors in any level of clustering, we can preserve spatiotemporal data neighborhood relationships in the TransGraph. The well-organized nodes in the TransGraph make it convenient for the user to judge the direction of time evolution and the correspondence between volume regions and graph nodes.

From Figure 6, we can observe that the TransGraph of the hurricane data set can be separated into two parts as indicted by the dashed line. These two parts of nodes actually correspond to the spatiotemporal regions of the hurricane's center and surrounding, respectively. This is verified by the rendering of four time steps corresponding to the nodes (states) selected in the TransGraph. We highlight the volume regions in their original color saturation to indicate their locations. As we can see, these two visually distinct spatiotemporal regions in the hurricane data are well separated in the TransGraph. Since our TransGraph organizes states in a hierarchical manner, the user can easily explore a finer level-of-detail and make finer state selection to narrow down the spatial regions of interest. Figure 6 (a), (c), and (d) are examples that demonstrate this capability of hierarchical exploration.

In Figure 7, we can observe that multiple branches in the TransGraph of the ionization data set spread out along different directions after a certain time step. To find out where these branches correspond to in the volume, we select nodes (states) at time step 118 and highlight the volume regions in their original color saturation. We also extract the corresponding state transition for each branch and overlay with the volume rendering for better observation. As we can see, these four branches correspond to different spatial regions and express their respective state evolutions.

5.3 TransGraph Query

Leveraging the TransGraph, we can perform state, transition, and time step queries to further understand the underlying data. In Figure 8, we show two examples of state and time step queries with the combustion data set. In (a), we query the first-level nodes (states) with the outgoing transition frequency larger than 0.108 and the query result is shown with colored nodes in the graph (nodes in the current time step are shown in red and blue, others are in green). We can see three branches of colored nodes along the direction of time evolution. They correspond to the upper, middle, and lower regions in the volume, as shown in (b). In (c), we perform a similar state query on the second-level nodes (states) and the brushed result is highlighted in (d). Comparing (a) and (c) as well as the rendering results in (b) and (d), we can infer that the middle part of the TransGraph (and the volume) has more nodes with larger incoming transition frequency than the upper and lower parts of the TransGraph (and the volume). In Figure 9, we show two examples of transition query. The first query in (a) shows transitions with frequency larger than 0.12. This is to bring out transitions that are frequently happening through the time series. The volume highlighting in (b) indicates that at time step 16, these transitions correspond to the middle region in the volume—the place where the two layers interact with each other. The second query in (c) shows nodes that are only involved in self-transition at time step 121. The volume highlighting in (d) indicates that those regions are stable in the sense that they only transfer to the same states in the next time step.

The TransGraphs in Figures 8 and 9 also tell us that in early time steps, we have a less number of states available. The number of states increases as the time step increases. Meanwhile, an increasing number of edges (transitions) exist among the upper, middle, and lower regions in the volume. This indicates the increasing turbulent nature of the combustion data set as the time evolves. Furthermore, the time step query in Figure 8 (c) which shows the number of active nodes in the second level also indicates the increasing number of nodes as the time step increases. It is important to point out that all these findings (except the correspondence between volume regions and graph nodes) can be solely inferred from the TransGraph beforehand without the present of the volume view. The volume view can help us put the graph in the context, make connection to the spatiotemporal data, and confirm our findings derived from the TransGraph.

5.4 TransGraph Tracking

In Figure 1, we show an example of dynamic tracking with the earthquake data set. A spiral pattern is formed for the TransGraph due to the large number of time steps (599) involved. The TransGraph reveals that the earthquake data set consists of many more states and transitions in the early time steps than later time steps. This complies with the fact that the earthquake breakout is within the first 200 time steps. After that, the earthquake diminishes gradually for the rest of 400 time steps. To perform dynamic tracking, we first select a volume region at time step 34, which corresponds to the earthquake's epicenter and the region is highlighted in its original color saturation in (a). The tracking results on the volume and graph are shown in the figure. Even though

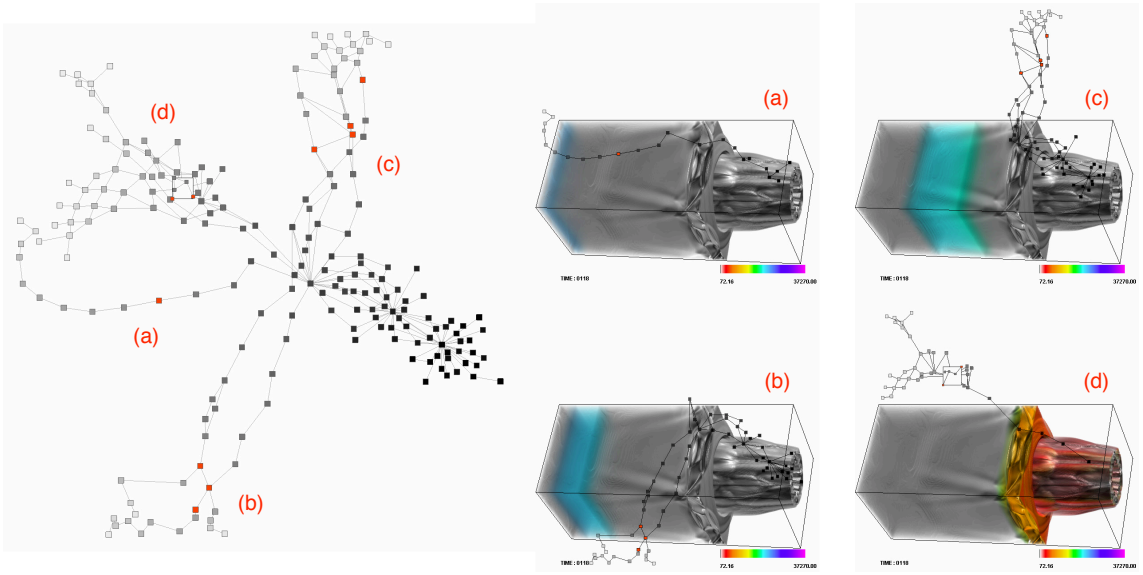
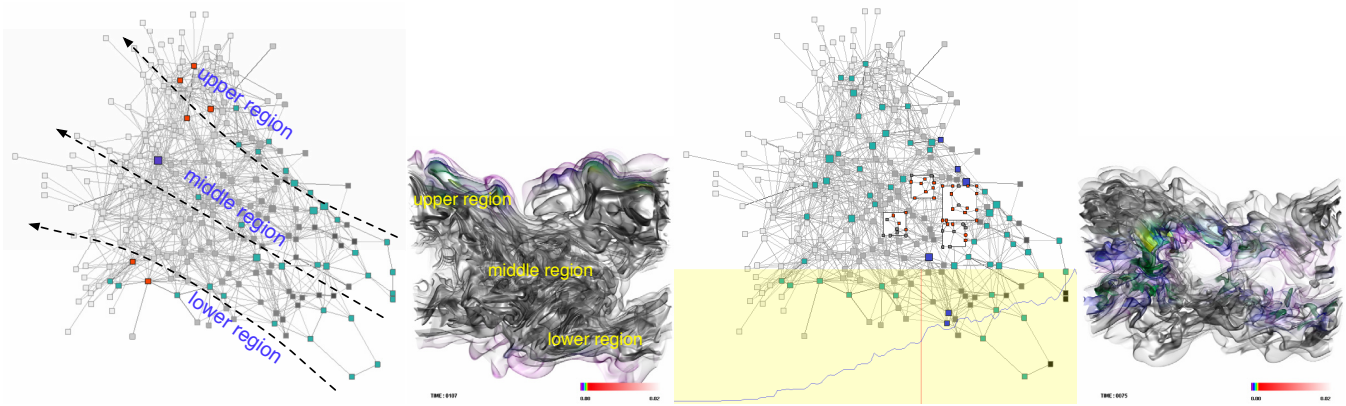


Fig. 7. Left: the TransGraph of the ionization data set. Right: (a) to (d) are the corresponding volume highlighting indicating the respective red nodes selected in the TransGraph at time step 118. Each spatial region corresponding to the states in red is tracked over time and the state evolution is extracted from the full graph. These separate spatial regions highlighted correspond to different branches in the TransGraph.



(a) (b) (c) (d)

Fig. 8. The TransGraph of the combustion data set with state and time step queries. (a) the query of the first-level states having their outgoing transition frequency larger than 0.108. These nodes are in red, blue, and green. (b) the volume highlighting corresponding to the red nodes brushed at time step 107. The node that is not selected is in blue and the rest in other time steps are in green. (c) the query of the second-level states having their incoming transition frequency larger than 0.057. (d) the volume highlighting corresponding to the red nodes brushed at time step 75. The time step query, displayed in the lower part in (c), shows the trend of the increasing number of active nodes in the second level over time.

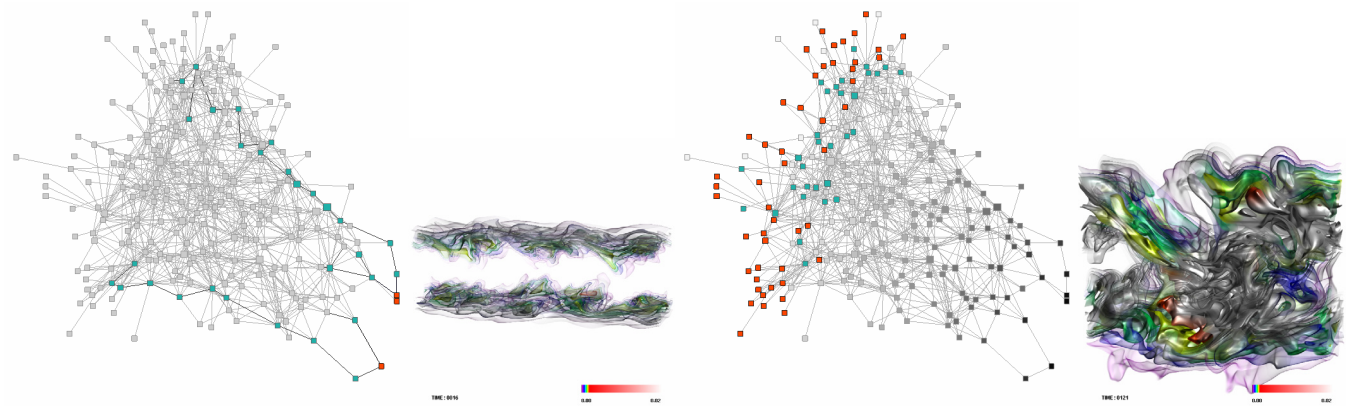
the underlying transitions are computed on a block-wise manner, we allow the user to adjust the propagation speed to give the impression of voxel-wise data transition. Specifically, for each of the initial blocks selected for tracking, we propagate its transitions along 27 neighboring blocks (including itself, i.e., self-transition) with a constant speed. The transition probability for each pair of blocks is precomputed and we use the average transition probabilities over a moving time window to ensure smooth transition of color saturation over time. Moreover, we consider the distances of a voxel to the centers of initial blocks selected to further adjust the color saturation of the voxel. The result in Figure 1 (d) shows the effect of anisotropic transition as the color saturations of purple voxels are not the same even though they are equally close to the initial blocks' centers.

5.5 Parameter Choices

Figure 10 shows the TransGraph layouts of two data sets under different parameter settings. Tables 1 and 2 list the detail of parameter values used. In Figure 10, the nodes of the TransGraph are colored according to the number of children in the next level of the hierarchy, modulated by the order of the time steps (early time steps darker, later time steps brighter). Comparing Figure 5 with Figure 10 (a) and (b), and Figure 1 with Figure 10 (c) and (d), we can see that although different sets of parameters give different graph layouts, the topology of the TransGraph remains almost the same and is insensitive to parameter changes and the randomness of the Fruchterman-Reingold algorithm. Therefore, the TransGraph is largely determined by the nature of the underlying time-varying data, while parameter changes only introduce slight variations in the layout.

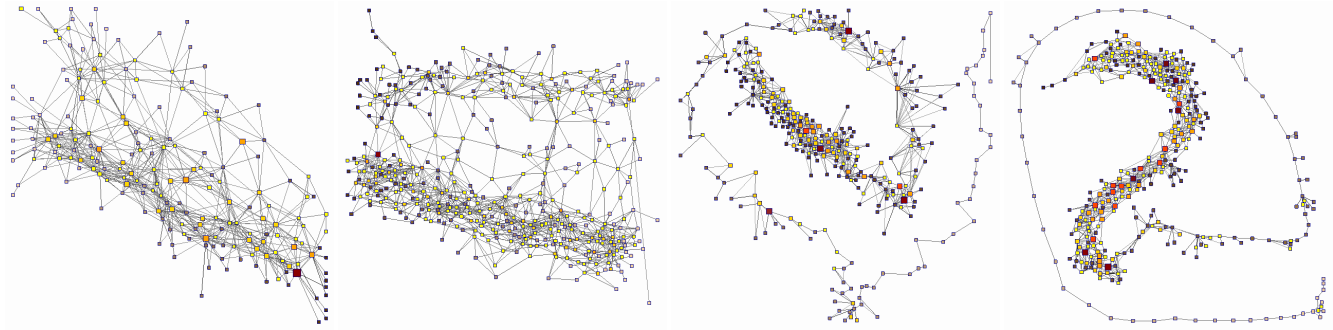
6 Limitations and Future Work

Our work has the following limitations. First, our definition of state is based on the data blocks and their value distributions in the first place. The subsequent hierarchical state clustering also operates on data blocks. Therefore, we are only able to cluster states at the block level. Reducing the block size could help refine the clustering results, but at the expense of increasing the size of the TransGraph and affecting the efficiency of our approach. Second, our current solution considers a constant propagation speed in all directions, which works



(a) (b) (c) (d)

Fig. 9. Left: the TransGraph of the combustion data set with transition query. (a) the query of the first-level transitions (either outgoing or incoming) having their frequency larger than 0.12. These edges are in black and corresponding nodes are in red and green. (b) the volume highlighting corresponding to the red nodes brushed at time step 16. The rest of nodes in other time steps are in green. (c) the query of the first-level states that are only involved in self-transition at time step 121. These nodes are in red and the rest of states at the same time step are in green. (d) the volume highlighting corresponding to the red nodes shown in (c).



(a) (b) (c) (d)

Fig. 10. The TransGraph layouts with different parameter settings (refer to Tables 1 and 2). (a) and (b) are for the climate data set with block sizes of $15 \times 6 \times 9$ and $30 \times 11 \times 9$, respectively. (c) and (d) are for the earthquake data set with block sizes of $16 \times 16 \times 8$ and $32 \times 32 \times 16$, respectively.

out well for the blob-shape earthquake's epicenter as shown in Figure 1. There is a need of further investigation for a general tracking task. For example, one possible solution is to develop an anisotropic model that takes into account the global and local dynamic natures of each data set. Finally, although there is a general "flow" of time evolution in the TransGraph, the time could be structured more strongly in conjunction with better state or feature definition and tracking capability to support more easily understandable and accessible visual analysis. For instance, it would be ideal if our approach could track some of the important high-level features that emerge from the hurricane simulation, such as changes in eye trajectory and rotational wind speed, as the hurricane approaches landfall.

There are several directions we would like to pursue in the future. First, the CPU version of JSD computation was the bottleneck. But after a GPU version of JSD computation was implemented, it is clear that the CPU version of hierarchical state clustering became the bottleneck (refer to Tables 1 and 2). Therefore, we will seek a GPU implementation of hierarchical clustering to improve the overall performance. Second, we compute the distance between histograms of blocks or states for hierarchical clustering. Besides the data value, other quantities can also be derived to build multi-dimensional histograms for a more complete evaluation of block or state similarity. On the other hand, although convenient, the data do not have to be partitioned into blocks. A more precise solution is to partition the data based on features if known beforehand. Finally, we will extend our work to handle multivariate time-varying data sets. We can either simply compute joint histograms to account for a group of variables considered, or construct the transition relationships for every pair of variables separately. In the former case, much of our current work can be immediately reused. In the later case, we need to revise our TransGraph to allow showing simultaneously, both the relationships within a single variable and between a pair of variables.

7 Conclusions

We have presented the TransGraph, an approach to hierarchical exploration of transition relationships in time-varying data. The organization of our TransGraph allows the direction of time evolution easily readable and the correspondence between volume regions and graph nodes easily inferable through brushing and linking. The TransGraph augments our ability to better understand time-varying volumetric data by providing an occlusion-free overview map that encompasses all time steps, controllable interaction that helps users track data transition over space and time, and adaptive exploration that allows us to go beyond small- and medium-scale data sets. As the size and complexity of data continue to increase, the TransGraph has the potential to become a handy tool for us to explore data in a cost-effective manner.

Acknowledgments

This work was supported in part by the U.S. National Science Foundation through grant IIS-1017935, and Michigan Technological University through a REF Research Seed grant. We would like to thank Jacqueline H. Chen and Andrew T. Wittenberg for providing the combustion and climate data sets, respectively. We also thank the anonymous reviewers for their insightful comments.

REFERENCES

[1] H. Akiba and K.-L. Ma. A tri-space visualization interface for analyzing time-varying multivariate volume data. In *Proceedings of Joint Eurographics - IEEE VGTC Symposium on Visualization*, pages 115–122, 2007.

[2] H. Akiba, C. Wang, and K.-L. Ma. Aniviz: A template-based animation tool for volume visualization. *IEEE Computer Graphics and Applications*, 30(5):61–71, 2010.

[3] S. Bachthaler and D. Weiskopf. Continuous scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1428–1435, 2008.

[4] S. Bachthaler and D. Weiskopf. Efficient and adaptive rendering of 2-D continuous scatterplots. *Computer Graphics Forum*, 28(3):743–750, 2009.

[5] S. Bruckner and T. Möller. Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1468–1476, 2010.

[6] W. Cui, Y. Wu, S. Liu, F. Wei, M. X. Zhou, and H. Qu. Context preserving dynamic word cloud visualization. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 121–128, 2010.

[7] G. Daniel and M. Chen. Video visualization. In *Proceedings of IEEE Visualization Conference*, pages 409–416, 2003.

[8] S. Diehl and C. Görg. Graphs, they are changing. In *Proceedings of International Symposium on Graph Drawing*, pages 23–30, 2002.

[9] Z. Fang, T. Möller, G. Hamarneh, and A. Celler. Visualization and exploration of time-varying medical image data sets. In *Proceedings of Graphics Interface*, pages 281–288, 2007.

[10] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software - Practice and Experience*, 21(11):1129–1164, 1991.

[11] S. Guthe and W. Straßer. Real-time decompression and visualization of animated volume data. In *Proceedings of IEEE Visualization Conference*, pages 349–356, 2001.

[12] H. Jänicke, M. Böttinger, and G. Scheuermann. Brushing of attribute clouds for the visualization of multivariate data. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1459–1466, 2008.

[13] H. Jänicke and G. Scheuermann. Visual analysis of flow features using information theory. *IEEE Computer Graphics and Applications*, 30(1):40–49, 2010.

[14] T. J. Jankun-Kelly and K.-L. Ma. Visualization exploration and encapsulation via a spreadsheet-like interface. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):275–287, 2001.

[15] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989.

[16] T.-Y. Lee and H.-W. Shen. Visualization and exploration of temporal trend relationships in multivariate time-varying data. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1359–1366, 2009.

[17] A. Lu and H.-W. Shen. Interactive storyboard for overall time-varying data visualization. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 143–150, 2008.

[18] E. B. Lum, K.-L. Ma, and J. Clyne. A hardware-assisted scalable solution for interactive volume rendering of time-varying data. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):286–301, 2002.

[19] K.-L. Ma. Image graphs - a novel approach to visual data exploration. In *Proceedings of IEEE Visualization Conference*, pages 81–88, 1999.

[20] S. Mehta, S. Parthasarathy, and R. Machiraju. Visual exploration of spatio-temporal relationships for scientific data. In *Proceedings of IEEE Symposium on Visual Analytics Science and Technology*, pages 11–18, 2006.

[21] P. Muigg, J. Kehrer, S. Oeltze, H. Piringer, H. Doleisch, B. Preim, and H. Hauser. A four-level focus+context approach to interactive visual analysis of temporal features in large scientific data. *Computer Graphics Forum*, 27(3):775–782, 2008.

[22] N. Sauber, H. Theisel, and H.-P. Seidel. Multifield-graphs: An approach to visualizing correlations in multifield scalar data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):917–924, 2006.

[23] H.-W. Shen and C. R. Johnson. Differential volume rendering: A fast volume visualization technique for flow animation. In *Proceedings of IEEE Visualization Conference*, pages 180–187, 1994.

[24] J. R. Shewchuk. Triangle: Engineering a 2D quality mesh generator and delaunay triangulator. In *Proceedings of ACM Workshop on Applied Computational Geometry*, pages 203–222, 1996.

[25] B.-S. Soho, C. Bajaj, and V. Siddavanahalli. Feature based volumetric video compression for interactive playback. In *Proceedings of IEEE Volume Visualization Symposium*, pages 89–96, 2002.

[26] C. Wang, H. Yu, and K.-L. Ma. Importance-driven time-varying data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1547–1554, 2008.

[27] R. Westermann. Compression domain rendering of time-resolved volume data. In *Proceedings of IEEE Visualization Conference*, pages 168–175, 1995.

[28] M. Wohlfart and H. Hauser. Story telling for presentation in volume visualization. In *Proceedings of Joint Eurographics - IEEE VGTC Symposium on Visualization*, pages 91–98, 2007.

[29] J. Woodring and H.-W. Shen. Multiscale time activity data exploration via temporal clustering visualization spreadsheet. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):123–137, 2009.

[30] J. Woodring, C. Wang, and H.-W. Shen. High dimensional direct rendering of time-varying volumetric data. In *Proceedings of IEEE Visualization Conference*, pages 417–424, 2003.