Journal of Visual Languages and Computing 24 (2013) 402–418

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

Visualization and analysis of 3D time-varying simulations with time lines

Li Yu [a], Aidong Lu [a,*], Wei Chen [b]

[a] Department of Computer Science, UNC Charlotte, 9201 University City Blvd, Charlotte, North Carolina, NC, 28262, USA
[b] State Key Lab of CAD&CG, Zhejiang University, Hangzhou, 310058, China

ARTICLE INFO

Available online 13 August 2013

Keywords:
Time line
Virtual words
Feature description
Time-varying data visualization

ABSTRACT

This paper presents a time line visualization approach, which allows users to study temporal relationships through encoding their interested data properties to time lines with different shapes and locations. Specifically, our approach extracts key data features as virtual words and uses them to encode various data properties. The distributions of virtual words across time are further applied to study various temporal relationships by generating time lines, which renders sampled time steps as points and temporal sequence as a line. Our approach consists of the three following components. First, we select feature points and collect feature descriptors to build a space of data properties, where virtual words are extracted as representative vectors. Second, the virtual words are applied to characterize feature points and their distribution statistics are used to measure temporal relationships. Third, we demonstrate several methods to visualize time lines flexibly for different data visualization and analysis purposes. We present several case studies to visualize time lines for different data visualization and analysis purposes. Our time line visualization can be used for both summarization and exploration of overall temporal relationships. We demonstrate with examples that time lines can serve as effective exploration, comparison, and visualization tools to study time-varying datasets.

1. Introduction

The explosion of large-scale time-varying datasets has created critical challenges for scientists to understand them effectively. With new computing technologies, scientific simulations nowadays can easily produce huge data volumes for a significant long duration. While previous visualization techniques can be used to study individual time steps, users often find it impractical to explore complex temporal relationships when facing large-scale datasets. The main challenge comes from the variations of data features that range from static object shapes to temporal events, which are hard to describe or measure. Efforts have been made to analyze temporal events or data relationships for time-varying datasets from different perspectives [1–3]. Different from previous methods, this paper presents a new approach to generate meaningful time lines and demonstrates their usages on several exploration and visualization tasks.

One critical challenge for large-scale time-varying data visualization is to abstract the temporal evolutions to understandable visual forms at different scales. We have designed a time line visualization which handles large-scale volumetric data visualization from three aspects. First, complex temporal evolutions are abstracted as succinct time lines, which are easy for studying changes and extreme locations. Second, a statistical mechanism is incorporated to handle large-scale datasets with extremely long time duration at different detail levels. Third, individual time steps are analyzed separately,

* Corresponding author.
E-mail addresses: lyu8@uncc.edu (L. Yu), aidong.lu@uncc.edu (A. Lu), chenwei@cad.zju.edu.cn (W. Chen).

1045-926X/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jvlc.2013.07.004

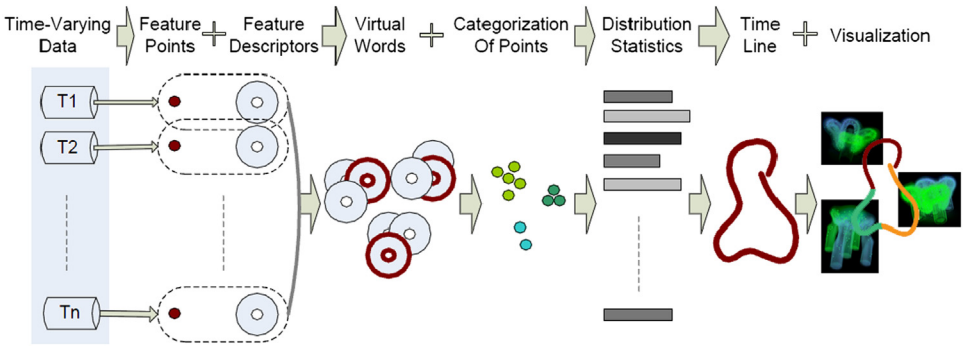

Fig. 1. Our approach consists of stages to select feature points and feature descriptors, extract virtual words and label feature points, and sample their distribution statistics across time to produce meaningful time lines for studying temporal relationships.

which can be easily accelerated with parallel processing and importance-based techniques.

Specifically, our approach is to treat temporal data as a collection of "*virtual words*". Different collections of virtual words can be used to represent data features from a time step; and transitions of virtual words can indicate temporal changes. As shown in Fig. 1, our approach consists of several stages. First, we select feature points by detecting the extrema locations at individual time steps of a time-varying volumetric scalar dataset and collect interested feature descriptors. Second, we extract a set of virtual words, which are in a flexible form to describe various data features, by performing clustering in the space of feature descriptors. Third, the distribution statistics of virtual words from different time steps are measured to produce meaningful time lines with a modified method of locally weighted bag of words (Lowbow) [4]. We present several time line visualization methods and demonstrate with examples that our approach can be used flexibly to explore and compare time-varying datasets. Case studies are provided to demonstrate that our approach can be used flexibly to explore and compare time-varying datasets.

The main contribution of this paper is a novel means of generating time lines based on extracted virtual words for characterizing data properties. Our approach to constructing virtual words provides a flexible framework for users to choose their desired data properties by combining a set of independent feature descriptors. We modify the Lowbow algorithm to suit the needs of time-varying data visualization and extend it to incorporate existing knowledge or hypothesis into the process of time line generation. Our approach ensures that the combination of time lines and feature descriptors provides a succinct visualization tool for data exploration, comparison, and summary when studying temporal relationships.

The remainder of this paper is organized as follows. We first review the related work of time-varying data visualization, scale-invariant feature transform (SIFT) [5], and Lowbow algorithms in Section 2 We then describe our approach to selecting virtual words in Section 3 and generating time lines in Section 4. Section 5 presents several methods to expand the concept of virtual words and visualize time lines for exploring and analyzing time-varying datasets. Section 6 provides example results, case studies and discussions on the usages of time lines for different visualization purposes. Finally, we conclude this paper and discuss future work in Section 7.

2. Related work

2.1. Time-varying data visualization

Time-varying 3D data visualization [6] has created many critical challenges due to the fast expanding size of time-varying datasets. A popular solution is to visualize temporal changes using time curves, which have been presented in several forms during the past years. Particularly, time-activity curve (TACs) [7] has been used to identify regions with similar temporal patterns through generating temporal functions for voxels in the volume and measuring their similarities. Later, Woodring and Shen [8] designed a global time view spreadsheet, which groups similar activities that are clustered using wavelet transform along time. Also, Wang et al. [2] presented an importance-driven method, which derived an importance curve for each data block based on the formulation of conditional entropy from information theory and clustered importance curves to visualize temporal trends.

These time curves are usually 1D time curves, which use horizontal axis to represent time and vertical axis to present a "property" of the dataset. In this paper, the time line is expanded to 2D space, where geo-locations are used to represent data similarities. Also, approaches time curves often requires a number of curves, as each curve only represent a data voxel or block. Our time line approach allows users to visualize dominating temporal trends with just one line.

Other techniques have been proposed with new visualization and comparison capabilities too [9–11]. For example, Sukharev et al. [9] studied data correlation through performing data clustering and segmentation using K-means and graph partitioning algorithms. Joshi et al. [11] proposed an illustration-inspired technique to visualize the structure and evolution of hurricanes. Lee and Shen [1] presented an algorithm called SUBDTW to identify trends appear and vanish during a time series. Cabane et al. tracked features over time by analyzing local textural properties and finding correspondent properties in sub-sequent volumes [12]. Kumar et al. used the frequency information of extracted features to color a bitmap. By visualizing the similarities of

these bitmaps using MDS, they can quickly discover clusters, anomalies and other regularities [13].

These methods are often designed to visualize or compare a small number of time steps. This is different from our goal of visualizing global temporal trends for general time-varying datasets. Especially when millions of time steps are involved, most of the methods require downsampling in space or in time. Our approach can flexibly visualize time-varying datasets in different sizes at various scales because of the statistical mechanism.

2.2. Feature extraction

In this paper, we include volumetric SIFT algorithm to select feature points for 3D datasets. The SIFT algorithm was originally proposed in image processing community to detect and describe local features in images. It has been applied successfully to different applications, such as object recognition [5], point tracking [14], panorama creation [15], medical imaging [16,17], and knowledge-assisted visualization [18].

Some approaches of feature tracking are also related to this topic, since they can provide the frame-to-frame correspondence between objects-of-interest to reveal the temporal trend of a time-varying dataset. The tracking information can be further studied to detect significant data changes. Currently, most feature tracking approaches are based on pre-defined feature models or user-specified regions-of-interest. The matching of data features is generally achieved by the following two mechanisms. First, based on selected regions-of-interest for feature tracking, either data features are matched based on their corresponding positions [19] or topological features are tracked using high dimensional geometries [20]. Critical points of geometry models have also been studied in many applications [21–23]. Second, feature attributes, such as position and size, are derived from data models and used to measure data changes. For example, Samtaney et al. [24] introduced several evolutionary events and tracked 3D data according to their feature attributes. Banks and Singer [25] used a predictor-corrector method to reconstruct and track vortex tubes from turbulent time-dependent flows. Reinders et al. [26] matched several attributes of features and tracked feature paths based on the motion continuity.

We extend the design of feature descriptors to a set of independent local data properties. More importantly, we derive our concept of virtual words from feature descriptors as representative key local data properties.

2.3. Locally weighted bag of words

Another important algorithm used in this paper is Locally weighted bag of words(Lowbow). Lowbow was originally presented for document visualization using curves [4]. By treating a document as a discrete categorical time series dataset and locally averaging a word histogram at different location in the document, a local version of the global histogram can be obtained to describe local word distribution. By viewing the histograms geometrically, a smooth curve is generated which summarizes the progression of the semantic and statistical trends within the document. Similarly, Assa et al. [27] presented human motions in succinct line drawings by selecting key poses based on the analysis of a skeletal animation sequence. We modifies the Lowbow algorithm for time-varying data visualization and keeps the advantages of Lowbow algorithms on sampling data from distribution statistics. Matthew et al. [28] proposed a similar technique, mapping the temporal information into a glyph and then positioning the glyph via PCA in a shape space. In this way, the visualization can reveal a wide variety of features in the data, including cycles of varying duration and anomalies and trends at different scales.

3. Generation of virtual words

To explore various temporal relationships, our approach is to treat data as a collection of virtual words, which describe key data features. This approach allows us to visualize large-scale time-varying datasets effectively through statistically analyzing the distributions of virtual words/data features. Since the set of virtual words is significantly smaller than the size of the original dataset, the performances of analysis and measurements can be greatly improved. What's more, the statistical aspect of our approach allows us to handle extremely large datasets with millions of time steps, which is impossible for direct comparison methods.

We define “virtual words” as data properties, in the form of high-dimensional vectors, at feature locations in the 3D space. Therefore, the first two steps of our approach are to select feature locations and to collect feature descriptors. We build our approach based on the SIFT algorithm, as it provides robust feature tracking which is invariant to rotation, scaling, noise or changes in illumination. These characteristics make it appropriate for extracting features from volume datasets. Later, we describe our method to generate a set of virtual words from the collected information. The following describes the details of these three stages respectively.

3.1. Selection of feature points

The first step is to select special 3D locations in a volume as feature points. While the definition of feature points varies, we select feature points through combining two criteria: one is the local extreme locations from volumetric SIFT algorithm [29], and the other is boundary points. By selecting feature points independently for each time step, this approach potentially allows different selection criteria or feature extraction methods to be applied for different time steps.

For 3D scalar datasets, we use volumetric SIFT algorithm to select feature points with three main steps: scale-space establishment, extrema location detection, and point filtering.

Scale-space establishment: We first build a scale space of difference-of-Gaussian (DOG) as follows. The original volume is scaled at several different levels: $\times 2$, original size, /2, /4, etc. Similar to the method introduced in [16], each of these volumes is convolved with a set of 3D Gaussian filter $G(x, y, z, k^p\alpha)$, where $p = [0, 1, 2, \ldots, n]$, to

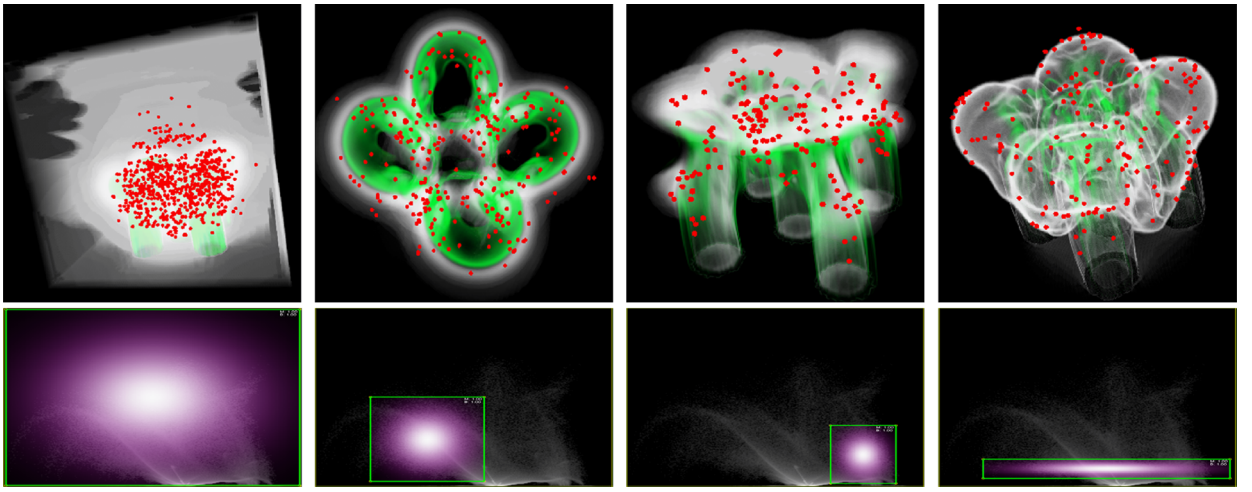

Fig. 2. Selection of feature points (shown on the first row) according to the choices of transfer functions (shown on the second row).

generate a scale space. A volume pyramid is built by subtracting the differences between volumes generated with adjacent p values. The choice of k is relevant to the scale of data features, as larger k^n results in smoother effects. For all the results in this paper, we use 4 volume levels and 12 DOG scales ($n=12$), and $2^{1/12}$ for k. We further detect our extrema locations in the DOG space.

Extrema location detection: From this scale space, we automatically select a set of candidate points by detecting local extrema locations. Generally the DOG value of each candidate point is compared to all its neighbors, of which $3\times 3\times 3-1=26$ voxels in the same volume, and $3\times 3\times 3=27$ voxels in each of the two adjacent volumes on the same level in the scale-space. A location is selected as a candidate only if it is larger than all of its neighbors or smaller than all the neighbors. This location is considered as a feature candidate.

Point filtering: The third step is to filter out inappropriate candidates, such as points that are poorly localized within an edge or with low contrast. As described in [16], a poor candidate feature point has a high principal value along the edge direction but a small value in the perpendicular direction. We calculate the curvature values $\lambda_1,\lambda_2,\lambda_3$ using the method from [30], where $\lambda_1<\lambda_2<\lambda_3$. A candidate point is valid if λ_3/λ_1 is smaller than a user specified threshold.

We use 15 as the threshold of λ_3/λ_1 for all our results in this paper based on observation.

During the process of interactive visualization, users often choose some sub-regions, e.g., through adjusting transfer function. Ignoring this interaction may miss feature points on the object boundaries. We accordingly complement the volumetric SIFT algorithm by considering the influence of boundary locations. Specifically, when the transfer function is adjusted, we modify the volume pyramid by keeping only the selected regions. The other regions are not considered during the process of scale-space extrema detection. This simple modification ensures that all the feature points inside the selected regions are preserved and extrema locations on the object boundaries can also be detected. We accelerate the process of feature point selection by pre-computing all the feature points and building the volume pyramid.

Fig. 2 shows example results of interactive selection of feature points, where the 2D transfer functions of voxel data values and gradient magnitudes are used. The first column shows all the feature points when the entire histogram is selected. The other three images demonstrate the results of point selections when different transfer functions are determined.

3.2. Collection of feature descriptors

To describe various local data properties, we extend the feature descriptors presented in the original SIFT algorithm [5] from the following two aspects. First, we add the components of gradient orientation for rotations and the object center location for movements in the feature descriptor. Second, we add components from texture analysis measurements for describing local data statistics. As each component describes a different data property, our construction method guarantees the independency among all the components of feature descriptors. This allows users to combine these components freely to describe the data properties under exploration.

The advantage of the SIFT feature descriptor is that it is invariant to orientation, scale, and location of feature points. We utilize this advantage of the SIFT algorithm to generate multiple independent components of feature descriptors.

The gradient orientation is built as follows. Assume we represent the gradient orientation in 3D by two angles, θ and ϕ. For each feature point, we rotate its neighborhood region by the angle of $\theta=\phi=0$ when computing its feature descriptor. The rotation matrix is defined as

$$R=\begin{bmatrix}\cos\theta\cos\phi & \sin\theta\cos\phi & \sin\phi\\ -\sin\theta & \cos\theta & 0\\ -\cos\theta\sin\phi & -\sin\theta\sin\phi & \cos\phi\end{bmatrix} \quad (1)$$

With this matrix, we sample a $4\times 4\times 4$ sub-region around the feature point in the rotated neighborhood. At each voxel of this sub-region, we generate a feature

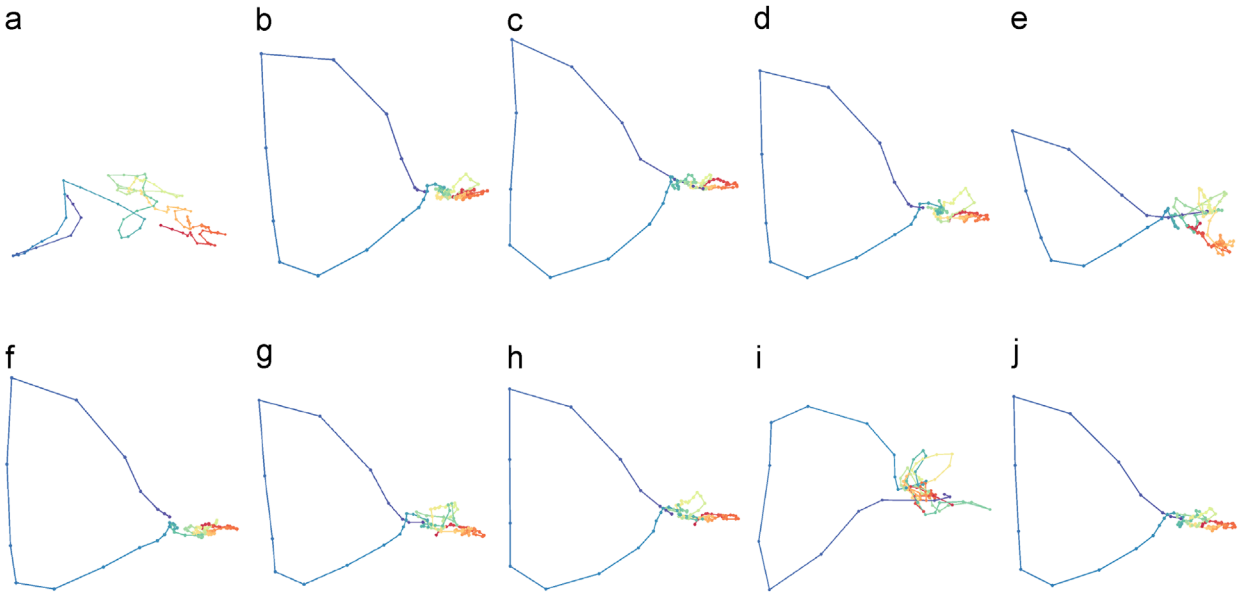

Fig. 3. Time lines generated with different feature descriptors for a time-varying energy simulation dataset. Overall, the time lines suggest two different data portions: the initial period (rendered in blue to cyan) and the end period (rendered in yellow to red). Different feature descriptions yield different shapes of time lines. (a) SIFT, (b) data value, (c) value hist, (d) value var, (e) mag, (f) mag hist, (g) mag var, (h) location, (i) orientation, (j) combined.

vector according to θ and ϕ. To quantify the value space, we use 8 bins for θ and 4 bins for ϕ. The gradient magnitude, weighted by a Gaussian window function centered at the feature point location, is added to the corresponding bin for the gradient orientation. In this way, the component of gradient direction has $4\times4\times4\times8\times4=2048$ dimensions.

We use the vector $(\theta,\ \phi)$ as the second descriptor component and the location vector (x,y,z) as the third descriptor component. This is necessary for time-varying data visualization, otherwise when an object moves or rotates across time, the time line will be just a point.

We have also explored other factors that can benefit time-varying data visualization. We mainly apply the first order of statistics for texture analysis to collect information of local data properties, such as variance and histogram [31]. Since this information is collected on the data after the rotation, it is independent to the gradient orientation. Obviously, it is also independent to location and time steps. Therefore, we can arbitrarily choose a combination of these descriptor components. The generation of time lines described below in Section 4 also allows users to select different weights for each descriptor component.

Fig. 3 shows the time line results from different feature descriptors for a time-varying energy dataset. Overall, the time lines suggest that the data changes gradually during the initial period (rendered in blue to cyan) and data properties are similar during the middle and end durations (rendered in yellow to red). We can see that the resulting time lines can be very different for these descriptor components, as they visualize different aspects of data properties. The first row shows the results of the SIFT descriptor, voxel value, histogram of value, magnitude of value, and gradient magnitude. The second row shows the results of histogram of gradient magnitude, variance of gradient magnitude, location, gradient direction, and the average combination of these nine descriptors. Therefore, it is important to provide users both the results of time lines and the choices of feature descriptors, so that they are aware of the meanings of time lines.

3.3. Extraction of virtual words

From the collection of feature descriptors, we extract virtual words as representative data properties. Here we use the "applicable components" of feature descriptors to refer to all the components except the location. Our first step is to cluster each applicable component of feature descriptors independently. This clustering step is supported by the fact that, similar to the color space, we can use a set of virtual words to describe the entire feature space. Also, the required number of virtual words is often limited, because it is bounded by the dimensions of feature descriptors. Specifically, we use K-means [32,33] since it is flexible to operate on any dimension. The choices of the clustering numbers should be different for various cases. From our experiments on both synthetical and real time-varying datasets, we find that the results are very similar as long as the cluster number is large enough. The statistical prospective of Lowbow algorithm used in Section 4 allows large sampling size. Therefore, we use 15 times the average number of feature points per time step as our clustering number. As shown in Fig. 4, The results of 1000 and 1500 clusters are similar and close to the real scenario. In this way, a set of virtual words is selected for each applicable component of feature descriptors.

Fig. 5 provide examples of automatically extracted virtual words. Since virtual words are vectors in the high dimensional space of data properties, they are hard to

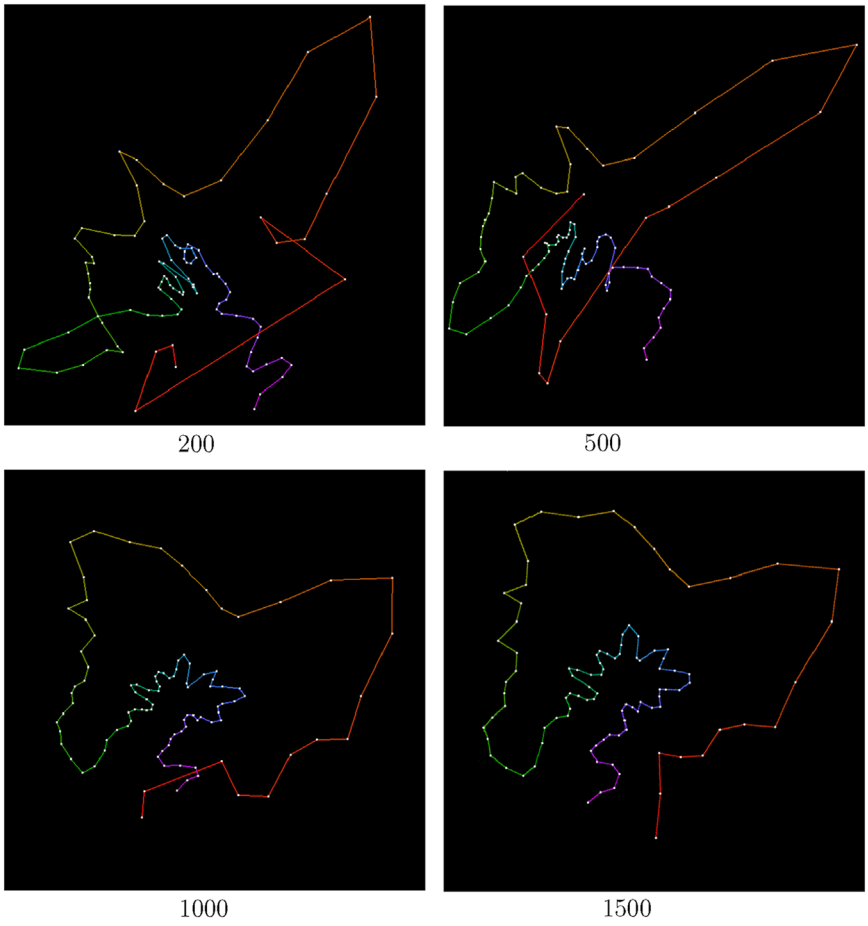

Fig. 4. Results with different clustering numbers. When the clustering number is large enough (1000 and 1500), the time lines results are similar.

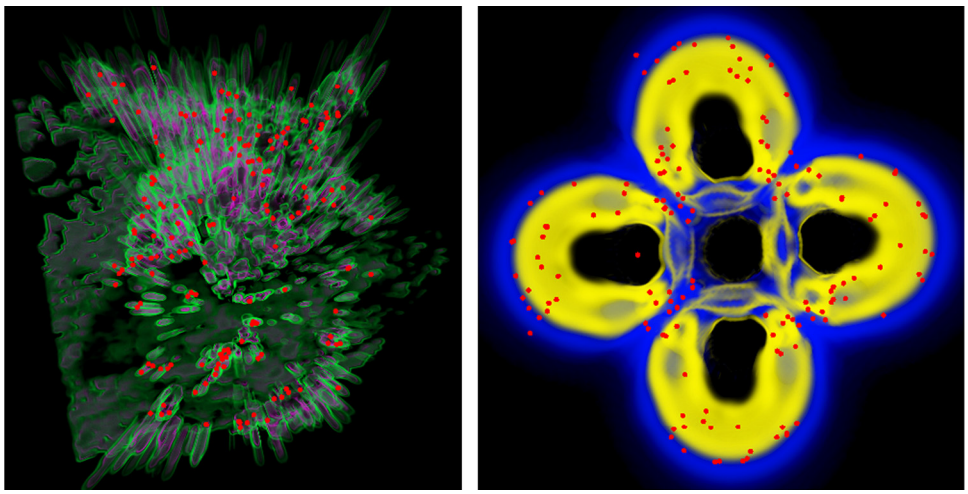

Fig. 5. Example of virtual words. They are feature descriptors located at the colored points.

visualize directly. We illustrate them in Fig. 5 by marking their locations in the volume. The counterparts of virtual words are the feature descriptors at these locations. The colors represent different virtual words. We can see that virtual words often appear at representative locations with large gradients or on the object boundaries. In Fig. 6, the automatically selected feature points are the eight corners of the cube. They are also the virtual words that describe

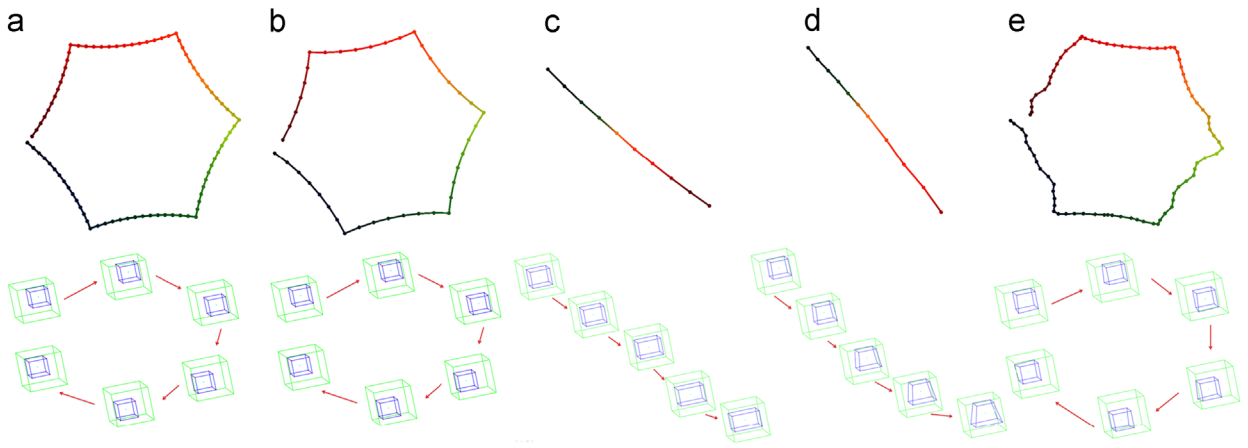

Fig. 6. Time lines can describe various types of events in time-varying datasets. Different Cube synthetic Datasets are used from (a) to (e): (a) and (b) cube moving around in the 3D space along the same path at different speed, (a) cube moving at 1 voxel per timestep, (b) cube moving 2 voxels per timestep. The densities of points in the time line reflect the speed of the movement. (c) scaling of the cube(the cube is expanding through the time), (d) deformation of the cube(the bottom half of the cube keeps changing position during the time), and (e) cube moving around under noises. The moving paths of (a), (b), and (e) are the same, which can be detected with time lines easily. The figures below the time lines illustrated the path of the movement.

several different types of temporal events. This result is consistent with our understanding of the cube dataset.

All the feature points are then categorized according to virtual words. For each applicable component of feature descriptors, we use the closest virtual word (determined by the distance in the space of feature descriptor) to represent each feature point. This is actually achieved from the clustering process simultaneously.

4. Creation of time lines

Since the virtual words are in the high-dimensional space of data properties, they cannot be visualized directly. Therefore, we transform the relationships among virtual words to time lines, which visualize time steps as points and adjacency relationships of time steps as lines. This transformation not only assists the analysis of temporal evolutions, but also allows us to sample time steps flexibly to satisfy the requirements of performance and accuracy. For this purpose, we adopt the mechanism from the Lowbow algorithm to sample time windows. Since the locations of sample time steps on a time line are calculated carefully based on data features, time lines can be used to analyze temporal relationships.

We achieve this goal with the following three steps. First, histograms are generated to record the distributions of virtual words for each time step. Second, we use these histograms to calculate dissimilarity matrices of different time steps. The distances are calculated based on Euclidean Distance. We also use time windows to group several timesteps into one histogram for sampling purposes. The resulting histogram of a time window can represent the distribution of virtual words within a time duration. Third, time lines are produced by feeding the resulting dissimilarity matrices to MDS method [34,35]. We project the Lowbow representation to the 2D plane, since 2D lines are more intuitive to understand.

As shown in Fig. 6, time lines can present a variety of temporal events, such as object movement, changing speed, object scaling, shape deformation, and effects under noise. The dissimilarities between each time steps are changing linearly in both scaling and deformation, therefore the changes of these two movements are similar. By observing the distance of points in the two time lines, users can see the changes of these two movements are different. We often use colors to indicate the temporal sequence. Most of our results use a blue to red colormap corresponding to the start to end of a time duration.

One advantage of Lowbow is the sampling feature. It allows us to sample time steps arbitrarily and emulate their histograms using different Gaussian windows. To ensure the smoothness of a time line, we can overlap adjacent time windows. For each applicable component i of feature descriptors, Lowbow is used to generate a dissimilarity matrix DM_i. We then combine all these matrices to a final dissimilarity matrix DM_{final} as the MDS input. Assume ST_1 and ST_2 are sample time steps. Here we allow users to assign weights $w(i)$ to the component i of feature descriptors, so that the effects of different data properties can be adjusted. When we assign larger weights for some descriptors, the combined time line is more affected by these factors.

$$DM_{final}(ST_1, ST_2) = \sum_i w(i) \times DM_i(ST_1, ST_2) \quad (2)$$

Specifically, for each applicable component of feature descriptors, one histogram is generated for each sample time step by collecting the distribution of all the virtual words in the sampling window. The values of each feature descriptor component are calculated independently for individual time steps. They are further collected for each sample time step. Some feature component only contains one value, such as the data value component used in Fig. 3 (the average data values of virtual words are calculated for each time step); while the other feature components are high dimensional vectors, such as the components of SIFT and histogram of data values. Let us define $\overrightarrow{f_i(t_p, v)}$ as the value of component i of virtual word v at time t_p. Then, we

can generate a dissimilarity matrix for the sample time steps by calculating the Euclidean distances between values $\overrightarrow{f_i(t_p,v)}$. The difference value $DM_i(ST_1,ST_2)$ of two sample time steps ST_1 and ST_2 is calculated using these equations:

$$DM_i(ST_1,ST_2)=\sum_{t_1\in ST_1,t_2\in ST_2}\sum_{v} dif_i(t_1,t_2,v) \quad (3)$$

$$dif_i(t_1,t_2,v)=|\overrightarrow{f_i(t_1,v)}-\overrightarrow{f_i(t_2,v)}|\cdot(num(t_1,v)+num(t_2,v)) \quad (4)$$

where $num(t_p,v)$ is the number of feature points correspond to virtual word v at time t_p and $\|$ represents the L2 norm.

For the location component, we calculate the average location of each virtual word from a user specified component of feature descriptors for each sample time step. An important component according to the data features under exploration can be selected for this purpose. For example, the SIFT component is chosen for the example in Fig. 3. The Equation (4) can be modified as follows:

$$dif_i(t_1,t_2,v)=|\overrightarrow{L(t_1,v)}-\overrightarrow{L(t_2,v)}|\cdot(num(t_1,v)+num(t_2,v)) \quad (5)$$

where $\overrightarrow{L(t_p,v)}$ is the average location of virtual word v at time t_p. The rest is the same as the applicable components.

To ensure the correctness of time lines for visualizing time-varying datasets, we need to handle the problem that different sample time windows may contain different numbers of time steps. If we simply follow the original Lowbow algorithm, the histograms for the sample time steps containing more feature points may collect information from smaller time durations, yielding an unequal property of the time lines. Therefore, we make the following changes to ensure equal window size everywhere. We can first calculate a histogram for each time step and normalize them using the numbers of feature points. Then, we operate sampling windows on time steps instead of feature points. All the applicable components of feature descriptors can be treated in this way. Similarly, the average location should be calculated for each time step, instead of every sampling window. This process results in an effect that sampled time steps may have different numbers of feature points, meaning that feature points from different time steps have different weights on the resulting time curves. We believe that this is a necessary change to keep consistent local statistics of data distributions for time-varying data visualization.

5. Time line visualization

This section describes several methods to visualize time lines for different exploration and analysis purposes.

First, an approach of parallel time lines is provided for simultaneous visualization of related information. Second, we describe a hybrid time line generation approach, which mixes input datasets, for comparing data from different time duration or attributes. Meanwhile, a focus+context method is employed to allow users to adjust the details of different time durations flexibly.

5.1. Parallel time lines

We explore parallel time lines to visualize the relationships between different data properties. Parallel time lines are generated by shifting the original time line along a certain direction on the 2D plane. This provides a visualization of multiple data properties simultaneously. It can be used to analyze the relationships between several data properties or different data fields across time.

We first determine the shifting direction to reduce the overlapping issue of parallel time lines. With the locations of sample time steps, we apply the principal component analysis (PCA) algorithm [36] to calculate the main directions of point distribution. The shifting direction is identified as the second (less important) direction of the PCA result. Second, we allow users to adjust the amount of shifts to achieve their ideal effect. Since the rendering parameters of a time line are very limited, we also explore controlling parameters of time line visualizations. For effective comparison, we need to limit the number of parallel time lines and the choices of different rendering parameters. For instance, the number of colors for visualizing different time lines can not be too large to ensure that the result is easy to recognize. Fig. 7 shows an example of parallel time lines for two types of information simultaneously. It is obvious to see the time durations when these data values are different.

5.2. Hybrid time lines for data comparison

To visualize the relationships between different time durations or data fields, we can also mix their datasets in the generation process of time lines. In this way, sample time steps from different data can be distributed on the same 2D plane according to their dissimilarities calculated from Lowbow. This allows users to compare their relationships across time closely. To distinguish different time durations or data fields, we generate separate time lines and render them with different parameters. Fig. 8 compares two pairs of different time durations (January 11–21 and 16–26) of attribute NH3 in an air quality datasets. We use different line

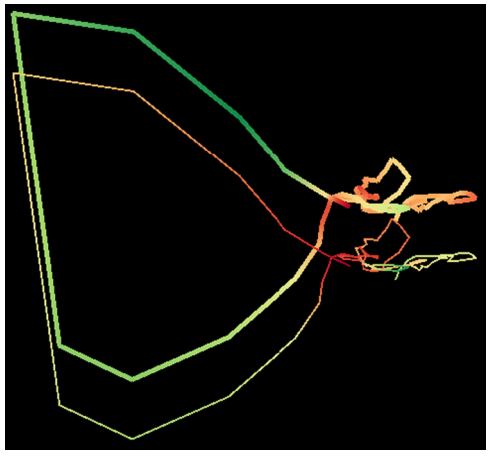

Fig. 7. We use parallel time lines to visualize the amount of feature points (the wider line) and average density (the thinner line) for an energy dataset. A green to red color map is used for values from min to max.

widths to differentiate the two time durations. The time line from (a) shows that the two time durations are similar, and the time line from (b) indicates that the two green portions in the middle are significantly different.

5.3. Multi-scale visualization

A multi-scale visualization is provided to users for adjusting the details of different time durations. We separate the

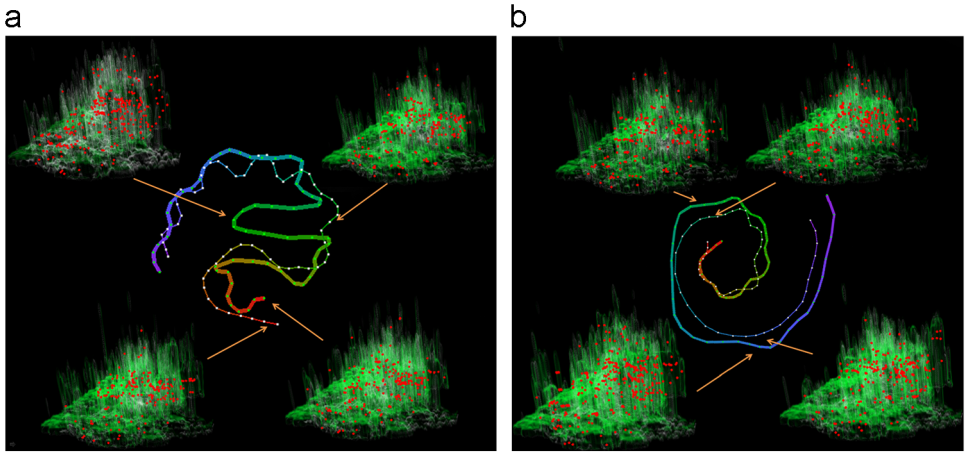

Fig. 8. With the time lines, we can easily compare different datasets or time durations. The snapshots from separate volume renderings provide details of the selected time steps.

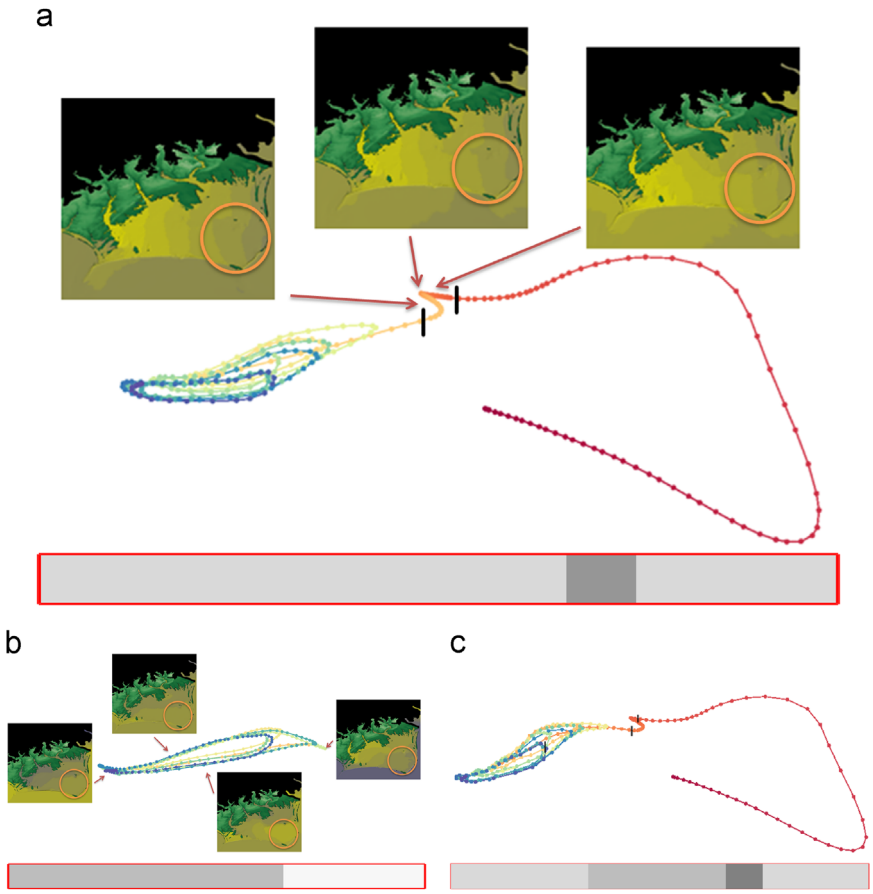

Fig. 9. Multi-scale visualization results of the storm surge dataset illustrated in Fig. 11. All the three subfigures (a)–(c) show time lines on the top and time importance degrees on the bottom. The colors of the time axis indicate the importance degrees of time steps, the darker the more important. Snapshots from user selected time steps are also provided.

visualization process into two independent steps: determining sample time windows and generating time lines.

In the first step, we initialize the importance values with 1 for all the time durations. Then, we update it according to user selections. Users can select the amount of sample time steps and adjust the importance values of different time durations respectively. With this information, we compute the number of sample time steps for each time duration, yielding an even distribution of importance of duration t as follows:

$$Avg_{im}(t) = n(t)/(imp(t) \times length(t)) \quad (6)$$

where $n(t)$ is the number of sample time steps, imp (t) is the importance value, and length (t) is the length of the time duration t. We then distribute sample time steps evenly within each duration.

To smooth a time line, adjacent time windows are overlapped. Generally we use five times the distance between two sample time steps as the size of time window. This parameter can also be interactively adjusted. The second step is to generate a time line with a histogram from each sample window. Time durations with larger importance values obtain higher values of Avg_{im}, thus revealing more details of the time durations.

Fig. 9 shows three time lines from user interaction. The time line (a) enlarges the time duration in the middle around the extrema location in orange. The time line (b) enlarges the portion in blue to yellow to observe the subtle details of the normal periodic trend. The time line (c) demonstrates that our approach allows flexible selection of important durations. By increasing the number of sample time windows, we can see more details in the time lines, reflecting more accurate information of the temporal durations.

6. Results and discussions

In Figs. 8, 10, 11, 12, and 15, the time lines are produced by our approach; the snapshots and arrows are added manually for illustrating selected time steps. Any other volume rendering method can also accompany our time line approach to visualize individual time steps.

6.1. Case studies

In the following case studies, we apply time lines to understand the major temporal relationships in the time-varying datasets.

6.1.1. Test case

To demonstrate the effects of time lines, we concatenate three different datasets, air quality, flow tube, and energy, to compose a time-varying dataset. As shown in Fig. 10, the automatically generated time line clearly shows three time segments with different colors. In this test, the differences between the three datasets dominate the shape of time line, resulting in three clusters on the time line. When we focus on the main temporal change by using a small number of sample time steps, shown in Fig. 10 (b), we can identify the three time segments easily. Larger numbers of sample time steps can be used for revealing more details of each time duration, as shown in (a) and (c).

6.1.2. Storm surge data visualization

We apply the time line approach to study the storm surge data simulation of hurricane Isabel. The dataset we use in this case study contains information around the sea shore at North Carolina. All the snapshots in Figs. 9 and 11 are colored in the same way: green is used to render

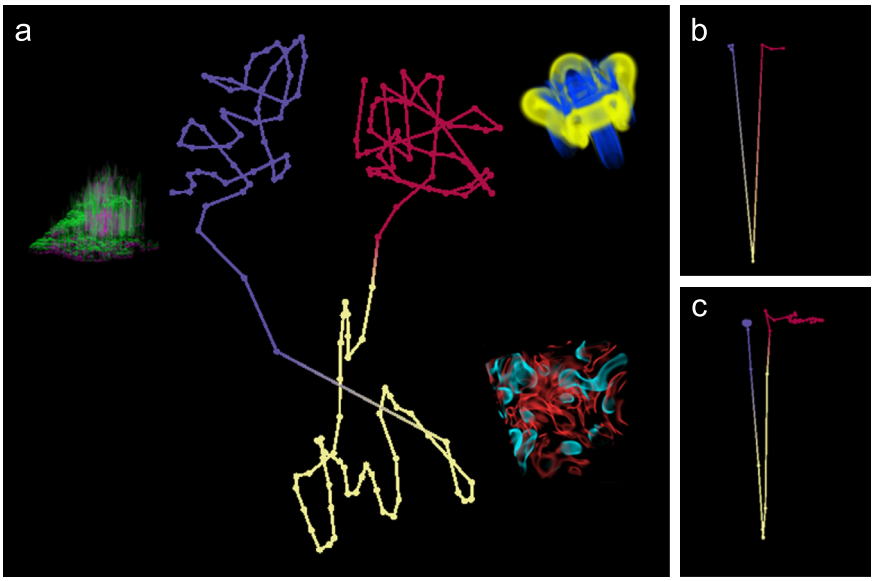

Fig. 10. Three time durations can be identified easily with the time line visualization. We generate three time lines by choosing different parameters (numbers of sample time steps, window sizes) for exploring different details: (a) (150, 10), (b) (60, 30) and (c) (90, 10).

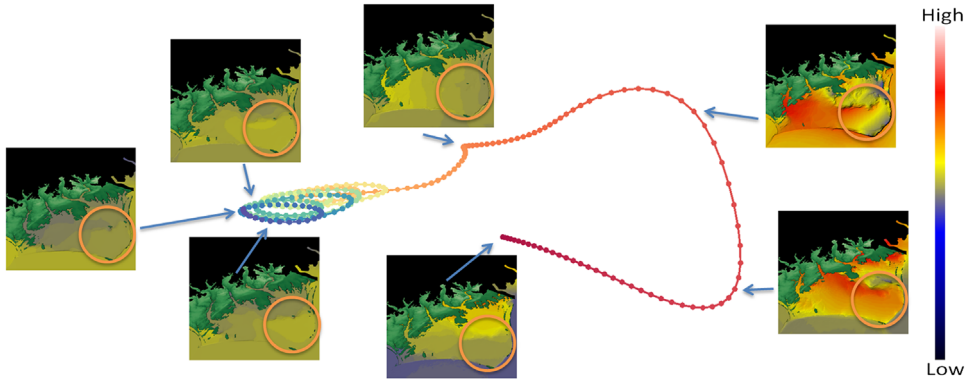

Fig. 11. The time line of a storm surge dataset demonstrates the effects of Hurricane Isabel to Outer Banks, North Carolina. The left portion (blue to yellow) corresponds to the periodicity of tidal elevations that influence every 12.4 h. The right portion (orange to red) indicates significant data changes of ocean surface from its normal distribution during the hurricane. The circle in the snapshots indicates the user specified regions.

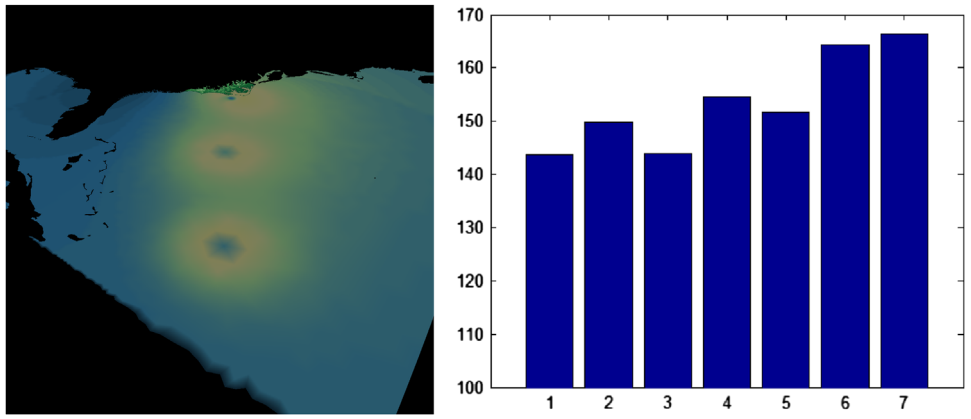

Fig. 12. (Left) Hurricane path is shown with three example time steps. (Right) The average heights of ocean levels are increasing before the hurricane, which confirms the shifts on the time line.

terrain and the rest colors of blue to red represent the heights of ocean.

As shown in Fig. 11, the time line visualizes several important temporal trends, which are difficult to detect otherwise.

First, we can detect a normal periodic feature (blue to yellow) and the effect of hurricane (orange to red), which corresponds to the abnormal high ocean levels in red as seen from the two snapshots on the right. Second, the time line suggests the starting time step of hurricane automatically with the extrema curve location in orange. Around the starting point, we can see a waiting period before the abrupt of hurricane according to the dense point distribution. Third, the distance between adjacent sample time steps increases, which indicates that the data changes dramatically during hurricane. Fourth, the linear curve shape of the hurricane duration is consist with the movement of hurricane center roughly in a line, shown in Fig. 12 (left). This movement event is similar to the cube examples in Fig. 6 (a), which can be identified from the line shapes as well. Fifth, the direction of the red portion at the end indicates that the data changes back toward its normal distribution after hurricane, as it moves toward the periodic cycles.

We can continue to explore the details of temporal trend by enlarging selected time durations, such as the periodic cycles shown in Fig. 9. As the cycles moves toward the hurricane portion, we can expect that the ocean levels are increasing gradually, which is confirmed with the average height per cycle measured from the data, shown in Fig. 12 (right).

6.1.3. *Air quality data visualization*

We also apply time line in the study of air quality with multi-field time-varying datasets. We use a continuous air quality simulation from the CMAQ model [37] in 12 months, of which each location takes 25 attributes. All attributes are collected 25 times every day. The data attributes measure the density of different chemicals.

We first generate time lines of four attributes respectively to visualize the data changes in a year. The number of sample time steps for this figure is 365 and the window size is 50. All the four time lines in Fig. 13 are colored from blue to red, representing Jan to Dec. Each time line

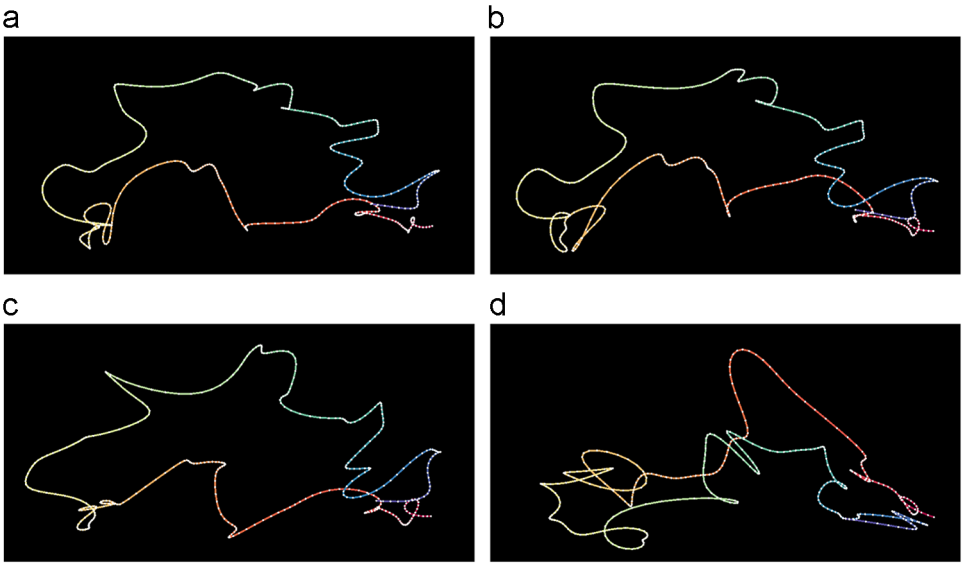

Fig. 13. Time lines for four attributes of an air quality dataset for a whole year suggests that NH3 is very different from the other three attributes. (a) NO, (b) NO2, (c) SO2 and (d) NH3.

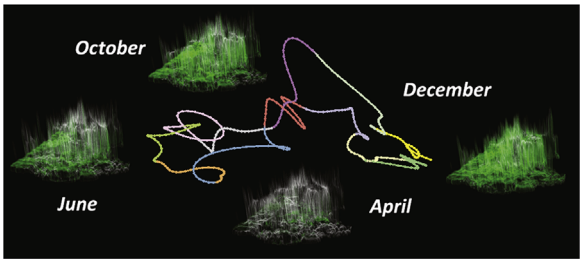

Fig. 14. To further inspect attribute NH3, we use 12 different colors to represent 12 months in one year.

visualizes an important data property: pollution volumes in the early spring and late winter are similar (the blue the red segments), and are different in the summer and fall (the green to yellow portion). This property is confirmed by domain scientists. Also, simply taking the time lines generated independently, we can detect that attributes NO and NO2 are similar, while NH3 is very different. This finding is consistent with the CMAQ model, since NO2 is generated according to the amount of NO.

Further, we focus on the special attribute NH3 and apply different colors to represent the 12 months in a year. As shown in Fig. 14, we generate the average 3D data visualization to explore the temporal change according to the time line. The four selected months provide visual contexts to understand that the pollution volumes change from the white and green bodies back and forth in a year.

We can further zoom into a smaller time range to study detailed changes. For example, we explore the attribute NH3 in June with a time line shown in Fig. 15. In this time line, we increase the number of sample time steps to 180 and window size to 25, so that daily-level temporal evolutions can be revealed. We also modify time line visualization by omitting the line segments between adjacent days, thus only the sample time steps belonging to the same days are connected. This helps users to study the temporal changes across different days, which are shown with the line structure patterns. Generally each line starts from outside, reach into the center at day, and drop back at night. Such line patterns indicate that the pollution volumes in the early morning and at late night of a day are similar; the volumes during day time in this month are similar. This finding is further confirmed with the 3D visualization, shown on the right of Fig. 15. This visualization reveals that the data variations of all the days in this month are similar. Therefore, users do not need to visualize all the time steps for every day in this dataset. This demonstrates that the data distribution in a time line can significantly shorten the exploration process for users.

We color the days from blue to red, so that we can tell June starts with the blue days and ends with the red days. The day colors do not show any obvious patterns, suggesting that the pollution levels at night are randomly distributed. This is also caused by the selection of feature points, which produces a large number of virtual words to describe various small evening features.

Fig. 15 (left bottom) shows the results for the same attribute in January and February. Comparing them with the time line of June, we can tell that the data changes are more irregular around these durations. The explanation from an air quality scientist is that the pollution levels are low in January and February, thus the data is more likely to be affected by random events. While around June, when the pollution levels are high, the main data trends play a more important role and thereby showing a more obvious line structure patterns.

6.2. *Quantitative results and discussion*

We run our algorithms on a PC with Intel Core2 CPU 6600 at 2.40 GHz and 2 GB RAM.

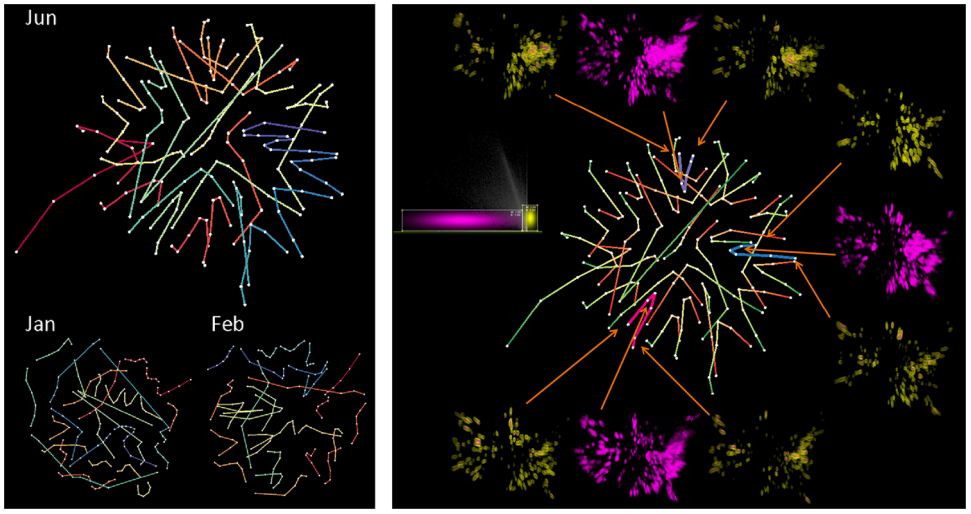

Fig. 15. Time line of attribute NH3 in June, colored by the number of days, shows obvious line structures. Every day starts at green and ends at red. And the middle of the day is located in the center of the figure. The snapshots shows the morning, noon and night renderings of three individual days. The three days are colored in pink, blue and purple.

The performance of our algorithm varies from the data size and complexity. First, the selection of feature points depends on the data size and the number of time steps. For the energy and air quality data, whose size is approximately 128^3, the time spent on this step is around 80 s. Second, the calculation of feature descriptors is linear to the number of feature points. All the feature components are very fast to compute, except the SIFT component, which can be accelerated with PCA-SIFT and parallel processing techniques. The clustering took several minutes. It took around 3 h to compute all the feature descriptors for the energy and air quality dataset. Third, the extraction of virtual words can be adjusted by the sizes of selected feature descriptors and the number of feature points, which can be accelerated with importance-based techniques. Fourth, the main algorithm used during the visualization process is the generation of selected time lines, which is less than a minute for the entire time-varying dataset.

Overall, the performance and memory requirements of our approach are linear in the number of time steps. The most time-consuming stages can be pre-processed.

6.3. *Discussion on visual design*

The efficiency of the time line visualization comes from the visualization of data relationships with 2D distances. Cognitive science has shown that human beings can effectively recognize object similarities from a representation by their distances [38]. Since the time lines distribute sample time steps on the 2D space according to their data features, users can easily identify some interesting events, such as periodicity, similarity, and important temporal events through extrema locations.

The major advantage of time line is its capability to incorporate the entire time-varying dataset into one single line to show its progression along the time. The modified lowbow algorithm allows user to show the temporal changes of dataset at different scales. For example, the user can display the time line of the air quality data on a monthly, daily or even hourly basis. This ensures that time line can handle long time sequence of data. Other approaches, based only on sampling through the entire dataset, will lead to information loss and inaccurate results.

Though our time line approach shows only the arbitrary shapes and does not have details as important regions inside each volume, the line is generated based on the features extracted from the dataset. It is easy for users to identify abnormalities and further explore the dataset to find events and features. It is impossible to visualize the dataset without any kind of data reduction when dealing with large-scale datasets. Without this step of information reduction, it will be more time-consuming to generate time lines. Note that, using feature points can still capture a wide range of data changes, particularly because we select feature points for each time step independently. When objects change significantly from previous time steps, new feature points often occur. Successful cases, like face recognition, have already shown that a few feature points can help reduce a large amount of irrelevant information.

Our case studies demonstrate that time lines are a very useful addition to rendering results for data exploration. Visualizing data corresponding to locations on a time line can help users understand the distribution and relationships of different time steps. With a time line, users can visualize just several time steps to obtain a quick understanding of the entire time-varying dataset. Our approach is more efficient than snapshots or animation, since it does not require rendering results from all the time steps. This is especially important for analyzing datasets with a large number of time steps. On the other hand, we realize that it may take time for users to get familiar with time lines. Our results of the cube dataset can be used as an example.

Generally, time lines are useful to visualize significant data changes. Small details may be lost among the overall temporal trend. Our approach allows users to choose transfer functions, select descriptor components, adjust the amount of feature points, and fine-tune level-of-details. These interactions can assist users to explore a complex dataset gradually.

6.4. Comparison

We compare our approach with four different methods. Among which, time histogram and time activity curves are popular approaches for time-varying data visualization. The last method is a different design based on our extraction method of virtual words.

6.4.1. Time histogram

Time histogram is a popular method to visualize the distribution of data properties across time. Generally, the horizontal axis represents time and the vertical axis represents data properties, such as data values.

The left image in Fig. 16 shows time histogram generated based on density values with 100 bins. The right one is generated using the extracted virtual words. Each column represents one sample time step and each bin represents one virtual word. There are 100 sample time steps and 500 words in the vocabulary.

Advantages of Time Histogram: Time histogram is simple to compute and straightforward for users to understand. It has also been applied widely in many fields. As shown in Fig. 16, the density histogram on the left indicates that the flow becomes stable during the second half of the time range. This is a great method when the data size is small and important data features are related to the voxel values directly.

Advantages of Time Line: Compared to time histogram, time line can visualize general temporal trends; while time histogram may miss many important temporal trends, as it only suggests the similarity of value distributions. Time line allows selection of data features beyond voxel values, which is crucial for exploring datasets with complex features. For datasets with long durations, time line can statistically sample the dataset at different scales; thereby producing visualization in a suitable resolution.

6.4.2. Time activity curves

Time curves are first used to reveal the temporal changes of time-varying medical visualization. Given a spatial place inside a body, the amount of activity measured is correlated to the biological function which is imaged. Many forms of time curves are then proposed and applied to scientific and volumetric datasets [7,8,2]. We have briefly described these approaches in the section of related work.

As shown in Fig. 17, the curves are generated based on L1 metric from Equation (1) in [7]. One hundred data blocks are used to render the time activity curves.

Advantages of Time Activity Curves: Time activity curve integrates the advantage of time histogram, which is intuitive to understand; and advantage of time line, which allows selection of various data features. Time activity curves work well for some applications, such as identifying functions of organs [7]. Since the location of organ is known before visualization, all the voxels within that spatial location can be clustered and calculated into one curve to represent the biological function.

Advantages of Time Line: First, time line does not have the clustering issue of time activity curves, which produce a number of curves for data blocks. Also, time line visualizes temporal trends directly; while time activity curves, the same as time histogram, require users to analyze temporal trends through similarity of curve values. Further more, time line represents similarity of two time

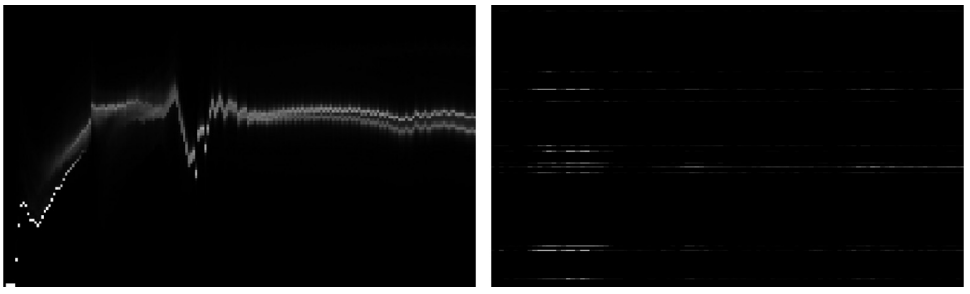

Fig. 16. Left: Time histogram of an energy dataset; Right: Virtual word histogram of energy dataset.

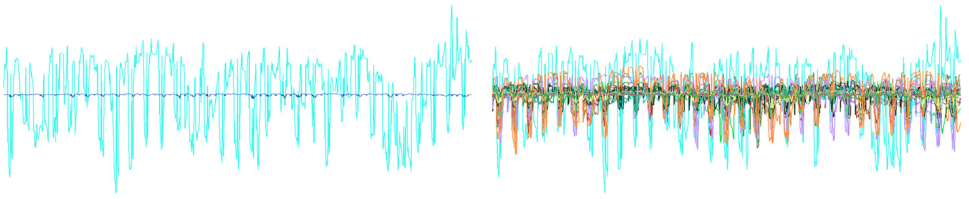

Fig. 17. Example time activity curves of several blocks of voles from air quality dataset. Left: only two curves are shown. Right: one hundred curves are shown.

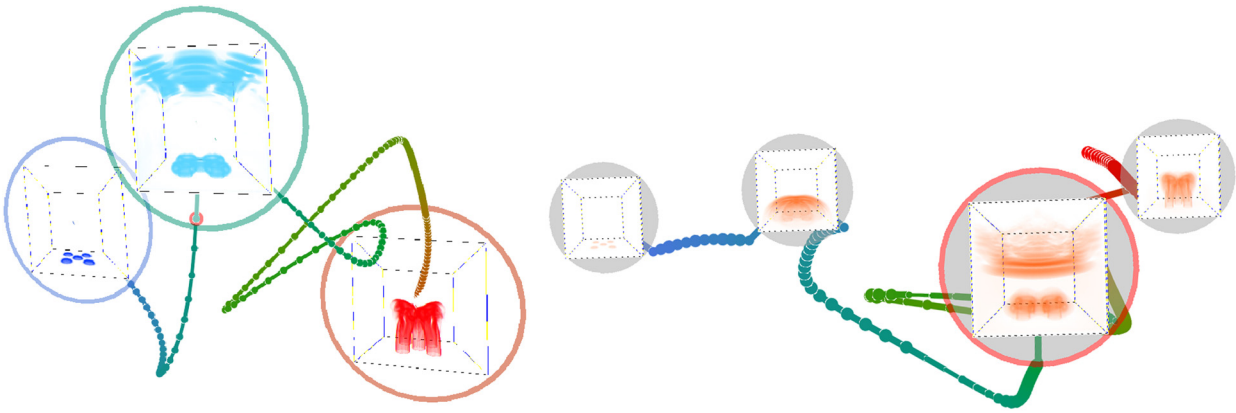

Fig. 18. Example visualizations of the energy data at two scales from the storyboard approach[3].

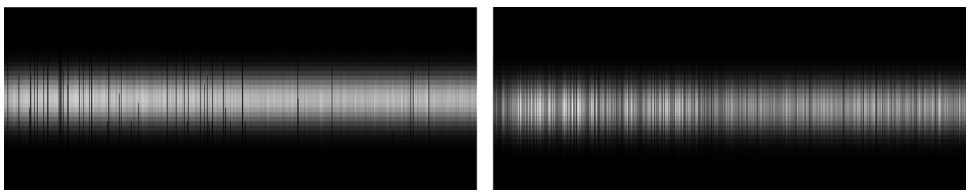

Fig. 19. Time window visualization of a hurricane dataset at early (left) and late (right) stages.

steps with distances on a 2D plan, which does not require users with professional trainings.

6.4.3. Storyboard

The closest previous work to this paper is the interactive storyboard approach [3] for visualizing overall data contents and relations of time-varying datasets.

Advantages of Storyboard: As shown in Fig. 18, the storyboard approach selects representative time steps from a time line and embeds their snapshots in the storyboard visualization. This design is convenient for users to understand the overall data contents and changes through both the example snapshots and changes of time lines.

Advantages of Time Line: First, the dissimilarity comparison of storyboard is based on data blocks. Every block in a time step is compared with the data block at the same location from all the other time steps, leading to n^2 (n is the number of time steps) performance. While time line can independently collect distribution of virtual words for each time window and compare distributions of each two time windows, leading to $n+h^2$ (h is the number of virtual words) performance. Since h is much smaller than n ($n \in [5,800]$ is generally much smaller than n), time line is more efficient than the storyboard approach. Second, the feature definition of storyboard approach only includes simple measurements such as data density, gradient and second order statistics. Time line allows feature selections based on representative voxel features/virtual words.

6.4.4. Visualization of time window

As seen in Fig. 19, bags of local virtual words are shown in histograms, which have time axis vertical and number of words horizontal. The histograms show how each small time duration are different from others. The brightness of each row reflects the amount of the particular virtual word inside the local time window. We can observe the differences between these two time stages.

Advantages of Time Window: The time window clearly shows how many different words are with in this time duration, which promotes the analysis of individual data features..

Advantages of Time Line: Time line is scalable to the number of time steps. Compared with time window, time line can handle large datasets with its statistical sampling component.

6.5. Discussion of choices

In our approach, we project the temporal differences to a 2D space with MDS; therefore, the time line is an approximation of the original difference measurement. This is unavoidable when representing a high dimensional data in the 2D space. Indicated by various studies that apply MDS [34,35], the MDS-based dimensional reduction scheme produces an optimal approximation of the temporal differences. Our approach, includes the measurement of temporal differences and later data visualization and comparison, can be integrated with other dimension reduction methods, such as PCA (Principle Component Analysis) [39], LLE (Local Linear Embedding) [40], LDA (Linear Discriminant Analysis) [41,42]. We choose MDS because of two reasons: first it is widely used in information visualization and graph visualization; second it is based on a matrix of similarities and assigns a location to each item in N-dimensional space, which matches our requirements exactly.

Some of steps in our approach can be simplified or replaced with other algorithms. For instance, we can use other feature extraction methods such as Harris Corner detection [43], SURF [44], or simple dissimilarity measurements like voxel values. All these methods are suitable for some types of data features. In particular, SIFT algorithm can capture gradient-based features and object boundaries very well, which are often important features for scientific datasets. We can adjust the parameters to concentrate on different types of features: for example, reducing the number of gradients in SIFT descriptor can point to features related to data values. The choice of the feature extraction methods should be finally determined according to the data properties and domain knowledge.

7. Conclusions and future work

In this paper, we present an approach to generate succinct time lines for visualizing the temporal trends of time-varying datasets. This approach is derived from the fact that human languages can describe various temporal events effectively with a limited set of words. For time-varying datasets, we characterize a set of virtual words according to selected data features within a data volume and treat the entire dataset as discrete descriptions with these words across time. In this way, many important temporal relationships can be detected through measuring the distribution of virtual words. We further summarize temporal relationships as time lines by projecting their dissimilarities to the 2D space. With provided interaction methods, our approach assists users to explore and analyze a time-varying dataset with a simple, yet effective tool.

We expect that our approach can be used for other types of datasets, such as time-varying vector fields and videos. We are interested in investigating solutions to improve the effectiveness of time-varying data visualization from the following aspects. First, we plan to expand virtual words as space-time feature descriptors, instead of corresponding to only a single time step, since they can represent the temporal trends directly. Second, we believe that the concept of virtual words can be used to improve the understanding of time-varying datasets in more flexible ways. We plan to design a multi-link interface with 3D rendering view, time line view and virtual words interactive views that can incorporate more information from virtual words to provide new visual analytics capabilities. Third, we are interested in embedding additional advanced comparison and analysis tools that use time lines as an interface to visualize time-varying datasets. Fourth, we are also interested in studying how noise in the original dataset affect the visualization results. We plan to use some quantitative measurements to show the confidence and uncertainty in our final visualization. Last, we plan to integrate some useful information we have collected from feature points and feature descriptors in our time line visualization. We are also interested in improving the low-dimensional embedding algorithms for time line generation.

Acknowledgement

We wish to thank the editors and reviewers for their valuable comments. This research is supported by DHS Center of Excellence - Natural Disasters, Coastal Infrastructure and Emergency Management (DIEM), DOE DEFG02-06ER25733, and NSF 0633150. This research is also supported by Major Program of National Natural Science Foundation of China (61232012,61202279), Zhejiang Provincial Natural Science Foundation of China (LR13F020001), Doctoral Fund of Ministry of Education of China (20120101110134).

References

[1] T.-Y. Lee, H.-W. Shen, Visualization and exploration of temporal trend relationships in multivariate time-varying data, IEEE Transactions on Visualization and Computer Graphics 15 (6) (2009) 1359–1366.
[2] C. Wang, H. Yu, K.-L. Ma, Importance-driven time-varying data visualization, IEEE Transactions on Visualization and Computer Graphics 14 (6) (2008) 1547–1554.
[3] A. Lu, H.-W. Shen, Interactive storyboard for overall time-varying data visualization, in: Proceedings of IEEE Pacific Visualization Symposium, 2008, pp. 143–150.
[4] Y. Mao, J. Dillon, G. Lebanon, Sequential document visualization, IEEE Transactions on Visualization and Computer Graphics 13 (6) (2007) 1208–1215.
[5] D.G. Lowe, Object recognition from local scale-invariant features, IEEE International Conference on Computer Vision 2 (1999) 1150.
[6] C. Johnson, C. Hansen, Visualization Handbook, Academic Press, Inc., Orlando, FL, USA, 2004.
[7] Z. Fang, T. Möller, G. Hamarneh, A. Celler, Visualization and exploration of time-varying medical image data sets, in: Proceedings of Graphics Interface 2007, GI '07, ACM, New York, NY, USA, 2007, pp. 281–288.
[8] J. Woodring, H.-W. Shen, Multi-scale time activity data exploration via temporal clustering visualization spreadsheet, IEEE Transactions on Visualization and Computer Graphics 15 (1) (2009) 123–137.
[9] J. Sukharev, C. Wang, K.-L. Ma, A. T. Wittenberg, Correlation study of time-varying multivariate climate data sets, in: PACIFICVIS '09: Proceedings of the 2009 IEEE Pacific Visualization Symposium, 2009, pp. 161–168.
[10] W. Aigner, S. Miksch, W. Müller, H. Schumann, C. Tominski, Visual methods for analyzing time-oriented data, IEEE Transactions on Visualization and Computer Graphics 14 (1) (2008) 47–60.
[11] A. Joshi, J. Caban, P. Rheingans, L. Sparling, Case study on visualizing hurricanes using illustration-inspired techniques, IEEE Transactions on Visualization and Computer Graphics 15 (5) (2009) 709–718.
[12] J. Caban, A. Joshi, P. Rheingans, Texture-based feature tracking for effective time-varying data visualization, IEEE Transactions on Visualization and Computer Graphics 13 (6) (2007) 1472–1479, http://dx.doi.org/10.1109/TVCG.2007.70599.
[13] N. Kumar, N. Lolla, E. Keogh, S. Lonardi, C. A. Ratanamahatana, Time-series bitmaps: a practical visualization tool for working with large time series databases, in: SIAM 2005 Data Mining Conference, SIAM, 2005, pp. 531–535.
[14] S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: spatial pyramid matching for recognizing natural scene categories, in: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, vol. 2, 2006, pp. 2169–2178.
[15] M. Brown, D.G. Lowe, Recognising panoramas, in: ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision, 2003, p. 1218.
[16] D. Ni, Y.P. Chui, Y. Qu, X. Yang, J. Qin, T.-T. Wong, S.S.H. Ho, P.A. Heng, Reconstruction of volumetric ultrasound panorama based on improved 3d sift, Computerized Medical Imaging and Graphics.
[17] W. Cheung, G. Hamarneh, n-sift: n-dimensional scale invariant feature transform, Trans. Img. Proc. 18 (2009) 2012–2021, http://dx.doi.org/10.1109/TIP.2009.2024578.
[18] J.E. Nam, M. Maurer, K. Mueller, Knowledge assisted visualization: a high-dimensional feature clustering approach to support knowledge-assisted visualization, Comput. Graph. 33 (5) (2009) 607–615.
[19] D. Silver, X. Wang, Tracking and visualizing turbulent 3d features, IEEE Transaction on Visualization and Computer Graphics 3 (2) (1997) 129–141.

[20] G. Ji, H.-W. Shen, R. Wenger, Volume tracking using higher dimensional isosurfacing, in: Proceedings of IEEE Visualization, 2003.
[21] T. Gerstner, R. Pajarola, Topology preserving and controlled topology simplifying multiresolution isosurface extraction, in: Proceedings of Visualization, 2000, pp. 259–266.
[22] B.-S. Sohn, C. Bajaj, Time-varying contour topology, IEEE Transactions on Visualization and Computer Graphics 12 (1) (2006) 14–125.
[23] H. Edelsbrunner, J. Harer, A. Mascarenhas, V. Pascucci, Time-varying reeb graphs for continuous space-time data, in: Proceedings of 20th Ann. Sympos. Comput. Geom., 2004, pp. 366–372.
[24] R. Samtaney, D. Silver, N. Zabusky, J. Cao, Visualizing features and tracking their evolution, IEEE Trans. Comput. 27 (1994) 20–27.
[25] D.C. Banks, B.A. Singer, A predictor-corrector technique for visualizing unsteady flow, IEEE Transactions on Visualization and Computer Graphics 1 (2) (1995) 151–163.
[26] F. Reinders, F.H. Post, H.J. Spoelder, Visualization of time-dependent data using feature tracking and event detection, The Visual Computer 17 (1) (2001) 55–71.
[27] J. Assa, Y. Caspi, D. Cohen-Or, Action synopsis: pose selection and illustration, in: Proceedings of ACM SIGGRAPH, 2005, pp. 667–676.
[28] M.O. Ward, Z. Guo, Visual exploration of time-series data with shape space projections, Eurographics / IEEE Symposium on Visualization (EuroVis) 30 (3).
[29] P. Scovanner, S. Ali, M. Shah, A 3-dimensional sift descriptor and its application to action recognition, in: MULTIMEDIA '07: Proceedings of the 15th International Conference on Multimedia, 2007, pp. 357–360.
[30] J. Goldfeather, V. Interrante, A novel cubic-order algorithm for approximating principal direction vectors, ACM Trans. Graph. 23 (1) (2004) 45–63.
[31] J.J. Caban, P. Rheingans, Texture-based transfer functions for direct volume rendering, IEEE Transactions on Visualization and Computer Graphics 14 (6) (2008) 1364–1371.
[32] D. Comaniciu, P. Meer, Mean shift: a robust approach toward feature space analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (5) (2002) 603–619.
[33] B. Georgescu, I. Shimshoni, P. Meer, Mean shift based clustering in high dimensions: a texture classification example, in: International Conference on Computer Vision, 2003, pp. 456–463.
[34] W. Torgeson, Multidimensional scaling of similarity, Psychometrika 30 (1965) 379–393.
[35] I. Borg, P. Groenen, Modern Multidimensional Scaling: Theory and Applications, Springer, 1997.
[36] S. Wold, K. Esbensen, P. Geladi, Principal component analysis, Chemometric and Intelligent Lab. Sys. 2 (1987) 37–52.
[37] ⟨http://www.cmascenter.org/⟩.
[38] B.M. Bly, D.E. Rumelhart, Cognitive Science (Handbook of Perception and Cognition), Academic Press, 1999.
[39] I.T. Jolliffe, Principal Component Analysis, Springer, New York, NY, USA, 2002.
[40] S. Roweis, L. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[41] R. Fisher, The Statistical Utilization of Multiple Measurements, Annals of Eugenics 8 (1938) 376–386.
[42] K. Fukunaga, Introduction to Statistical Pattern Recognition, 2nd ed. Academic Press Professional, Inc., San Diego, CA, USA, 1990.
[43] C. Harris, M. Stephens, A combined corner and edge detector, in: Proceedings of the 4th Alvey Vision Conference, 1988, pp. 147–151.
[44] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (surf), Comput. Vis. Image Underst. 110 (2008) 346–359, http://dx.doi.org/10.1016/j.cviu.2007.09.014.