Visualizing Time-Varying Features with TAC-based Distance Fields

Teng-Yok Lee* Han-Wei Shen†

The Ohio State University

ABSTRACT

To analyze time-varying data sets, tracking features over time is often necessary to better understand the dynamic nature of the underlying physical process. Tracking 3D time-varying features, however, is non-trivial when the boundary of the features cannot be easily defined. In this paper, we propose a new framework to visualize time-varying features and their motion without explicit feature segmentation and tracking. In our framework, a time-varying feature is described by a time series or Time Activity Curve (TAC). To compute the distance, or similarity, between a voxel's time series and the feature, we use the Dynamic Time Warping (DTW) distance metric. The purpose of DTW is to compare the shape similarity between two time series with an optimal warping of time so that the phase shift of the feature in time can be accounted for. After applying DTW to compare each voxel's time series with the feature, a time-invariant distance field can be computed. The amount of time warping required for each voxel to match the feature provides an estimate of the time when the feature is most likely to occur. Based on the TAC-based distance field, several visualization methods can be derived to highlight the position and motion of the feature. In addition, a vector field, called Feature Motion Field, can be derived to indicate the motion tendency of the feature in the local regions. Various flow visualization techniques can be applied to this vector field to create additional visualization. We present several case studies to demonstrate and compare the effectiveness of our framework.

1 INTRODUCTION

While creating animations is a standard method to visualize time-varying data, analyzing complex time-dependent features in detail via animations alone still presents several major challenges. For instance, due to the 2D nature of the computer display, comparing the size and shape and tracking the motion paths of small 3D time-evolving features can be difficult. Identifying complex temporal patterns in animations can be difficult for the users too since the users are required to infer quantitative information from images and to memorize the trends for multiple data points simultaneously. Finally, the transfer functions used in direct volume rendering (DVR) are often not designed to track data values of high dynamic ranges spanned across a long time sequence. As a result, important temporal features can be easily missed.

To overcome these problems, feature-based visualization techniques such as [15] [16] [13] [6] [1] [18] [5] and [2] have been studied in the past. With those techniques, not only a large data set can be reduced down to a smaller number of salient features, but also the complex dynamic of the time-varying data can be succinctly described by the features' characteristics in space and time. Most of the feature-based visualization techniques require extracting the features in the spatial domain and then tracking the evolution of the features by matching their positions over time. Feature extraction, nevertheless, can be nontrivial when the boundaries and the properties of the features are not well defined. The extraction and tracking of features can also be very time consuming.

In this paper, we propose a novel visualization framework to analyze time-varying data sets. A major goal of this framework is to allow the user to perform detailed analysis of a data set's temporal behavior, and to identify regions that exhibit different spatiotemporal properties. Specifically, we are interested in time-varying features that can be described as time series or called *Time Activity Curves* (TAC). TACs are common as feature representation in medical data, such as electrical signals of the heart via electrocardiography, brain signals via positron emission tomography (PET) imaging or functional Magnetic Resonance Imaging (fMRI), and electromagnetic radiation from dynamic SPECT. Previously, researchers have utilized TACs for analysis and visualization in various medical applications [20] [4] [3]. In scientific simulations, features of interest are also commonly represented as TACs. In an earthquake simulation, for example, the seismic wave measured at a fixed point over time forms a time series, whose shape can be used to determine the geometrical properties (e.g. the distance to the earthquake source) or the geological properties (e.g. the material) at the point.

To identify regions that exhibit a particular temporal trend represented by a user-specified feature TAC, we propose a novel method in our framework to convert the original time-varying volume into a distance field, called *TAC-based Distance Field*. The distance field stores the dissimilarity of each voxel's time series to the feature TAC, from which various visualization techniques can be applied to reveal the spatial distribution of the temporal feature. To measure the dissimilarity/distance between TACs, several issues need to be addressed. First, because the feature travels through space at a finite speed, different data points in space will encounter the feature at different times. This phenomenon can be observed as a shift of time in the data points' respective TACs. The width of the feature on the TAC can also be stretched/compressed because the feature may travel at different speeds in different regions. Another point of consideration is that as the feature travels in space, its property, for example the magnitude of the earthquake wave, may also gradually change over time. This will cause the shape of feature TAC to deform at the points along the feature's motion path. As an example, the three TACs shown in Figure 5 (a) illustrate how the feature TAC can change over time. It can be seen that the temporal patterns have different lengths, and the peaks appear in different positions inside the TACs.

Previously, L_1 distance metric, L_2 distance metric and cross-correlation were used to compare TACs, but these distance metrics cannot adequately address the issues mentioned above. To tackle the problems, we employ a metric called *Dynamic Time Warping* (DTW) to calculate the shift- and deform-invariant dissimilarity. DTW is a dynamic programming algorithm that aligns two time series with the smallest distortion. With DTW, TACs of similar shapes but with different temporal shifts and time spans can still be identified. Meanwhile, because DTW also provides a mapping between the time steps of two TACs, it is possible to estimate when the feature of interest emerges at different points in the spatial domain. As a result, the spatiotemporal revolution of the feature can be effectively depicted.

This paper is organized as follows. We review related works on time-varying data visualization and feature-tracking in Section 2.

*e-mail: leeten@cse.ohio-state.edu

†e-mail: hswhen@cse.ohio-state.edu

Detail about the TAC-based distance field is described in Section 3, including the TAC-based feature description, computation of DTW, and construction of the TAC-based distance field. The visualization algorithms are described in Section 4, including the design of transfer functions for volume rendering, and the derivation and visualization of the feature motion traces. Because the time complexity of DTW is quadratic to the input size, it is accelerated on GPUs, as described in Section 5. Section 6 presents the case studies using two time-varying data sets. Section 7 concludes this paper and presents possible future work.

2 Related Works

Visualizing time-varying data has been a focus of visualization research for the past decade. While there are several issues related to this research topic such as interactive rendering or efficient data compression, here we are mainly focusing on techniques that are targeted at effective displaying and tracking of time-varying phenomena.

2.1 Time-varying Data Visualization

To visualize time-varying features in illustrative styles, Post *et al* proposed a technique to render time-varying data as iconic symbols to represent the essential attributes of salient features [12]. In [17], Svakhine *et al.* proposed various techniques to visualize time-varying 3D flow or scalars by displaying the contour volumes with different visual enhancements. To render the motion of a time-varying feature, Joshi and Rheingans [7] draw the motion with conventional illustrative techniques such as speedlines to create visualization in different styles; this technique, however, requires a priori feature tracking stage to create the traces.

Several works have been proposed to visualize time-varying data sets without animation. In [21], similar to the Chronophotography technique, Woodring and Shen described an algorithm called Choronvolume to visualize data of multiple time steps in a single image. Another way to visualize time-varying 3D data is to consider the whole data set as a 4D volume, and then visualization can be achieved by rendering a 3D hyperplane in the 4D domain from different perspectives, as proposed by Woodring *et al.* in [22]. Compared to the previous works above, which are focused on visualizing isosurfaces or interval volumes, our framework provides a more general representation of the temporal features by using TACs with more rendering options.

Visualizing time-varying data by displaying salient features has been widely studied especially for large scale data sets since features require much less storage than the raw data. Feature-based visualization requires several components: feature definition, feature extraction and segmentation, feature tracking and visual representation of the features. In the next section several feature tracking algorithms are reviewed.

2.2 Feature Tracking

To track time-varying isosurfaces, Wang and Silver proposed several algorithms for regular grid [15] and unstructured grid [16] volumes. Their assumption is that there exists spatial overlap among corresponding isosurfaces in consecutive time steps. This assumption means that a series of overlapped isosurfaces is equivalent to a connected isosurface in 4D space. Ji *et al.* proposed an algorithm to take advantage of this property, where the tracking of 3D isosurfaces is reduced to generation and slicing of isosurfaces in 4D space [6].

To satisfy the assumption that there exists spatial overlap among corresponding features, the data needs to be sampled at a higher temporal resolution, which requires much larger storage space. To relax this constraint, different algorithms have been proposed. Reinders *et al.* proposed a tracking framework by predicting the variation of feature attributes, such as centroid, shape, size, orientation, etc. to estimate the temporal behavior of the features [13]. In [1], Bauer and Peikert transformed the input data into scale space, and then the non-overlaping features in the original scale can be tracked by searching in different scales.

One difference between feature tracking in scientific data and video data is that the features in scientific data can be merged from a set of features or split into disjoint features over time. By taking advantage of the fact that the corresponding sets of disjoint features often have similar shapes, Ji and Shen utilized the Earth Mover Distance [14] and decision trees to match sets of features without spatial overlap [5].

All of the related works above assumed that the feature segmentation before tracking can take place. However, except isosurfaces, segmentation of features may not be trivial. To track features in 2D unsteady flow, Theisel and Seidel [18] proposed a concept that transforms the original dynamic flow data into a steady 3D vector field called Feature Flow Field, where the tracking in the original domain is equivalent to creating streamlines in the new vector field. Joshi and Rheingans [2] proposed a texture-based tracking method by treating the time-varying volumes as 3D textures and tracking the features by mapping overlapped subvolumes with similar textural patterns.

3 TAC-based Distance Fields

Detail about the TAC-based distance field is described in this section, including the representation of features as TACs, the algorithm of DTW, and the construction of the TAC-based distance field.

3.1 TAC-based Feature Description

To allow a detailed analysis of time-varying data, it is necessary to separate the data points that demonstrate different temporal trends. Here we create a framework to allow visualization of time-varying features modeled as TACs. TAC-like features are commonly encountered in medical applications since the time-varying signals captured by medical imaging devices already can be thought of as a set of time series. For data from scientific simulations, sometimes additional transformation may be needed to convert the raw data into TACs. A vector field, for example, needs to be converted into a scalar field such as velocity magnitudes, from which the TACs can be created.

To specify a feature, the TAC can be manually edited by the users or automatically extracted from the input data. As an example, in our experiment we generated the feature TAC by first K-mean clustering the time series from all voxels. After the TACs have been grouped into distinct sets, one of the mean TACs from the K clusters is chosen as the feature of interest. The user can switch the TACs from one cluster to another and visualize the spatial distributions of different temporal features. This allows the user to perform detailed analysis of the time-varying data in a more systematic way.

3.2 DTW Distance Metric

Once the feature has been specified as a TAC, in order to detect the regions that exhibit a similar temporal trend, a distance metric is needed to measure the dissimilarity to the feature TAC at each data point. To design such a distance metric for two TACs, we first use DTW to align them with the smallest distortion and then use L_2 distance between the warped TACs as the final distance. We call this distance metric as the *DTW distance metric*. DTW is an algorithm to nonlinearly align two data sequences, which has also been used in some other disciplines [8]. As mentioned previously, the reason for us to employ the DTW distance metric is to account for the possible time shift and shape deformation that can occur to the TACs as the feature travels through space and time.

To compute the DTW distance between two data sequences, three constraints need to be considered, which results in a dynamic

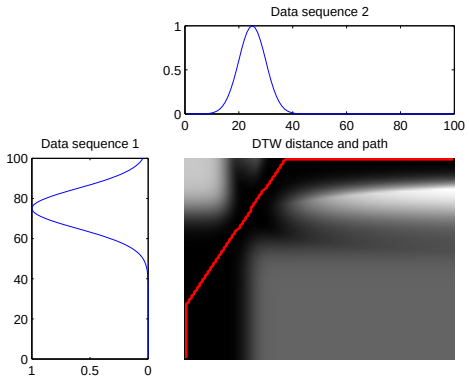


Figure 1: DTW between two synthesized data sequences. The two data sequences are normalized Gaussian functions with different means and variances. The warping is plotted as a red line in the lower right subfigure. The distance table is drawn as an image.

programming algorithm. First, the first and last data points in one sequence is always mapped to the first and last data points in the other sequence, respectively. Second, the warping should preserve the original order of data in the sequence. Third, two adjacent data points in the same sequence cannot be mapped to non-adjacent locations.

Based on these constraints, given two data sequences denoted as $x[1..m]$ and $y[1..n]$, DTW can be modeled as a recursive equation:

$$D[i,j] = dist(x[i],y[j]) + min\{D[i-1,j],D[i-1,j-1],D[i,j-1]\} \quad (1)$$

for $i = 2\ldots m$ and $j = 2\ldots n$, where $D[i,j]$ denotes the optimal distance after the warping between two subsequences $x[1\ldots i]$ and $y[1\ldots j]$ has been done, and $dist(\cdot,\cdot)$ denotes the distance function between two data items,which is the L_2 distance metric here. By storing the distances $D[i,j]$ for $i = 1\ldots m$ and $j = 1\ldots n$ in a 2D table D, the entries can be computed according to Equation 1 from $D[1,1]$ to $D[m,n]$ in scanline order. The warping can be then found by backtracking from $D[m,n]$ to $D[1,1]$. Figure 1 presents the result of DTW between two synthesized data sequences. The distances are plotted as a 2D image, and the warping is plotted as the red line.

The main benefit of DTW is that it can consider both the shifting and deformation of the time series, while other distance metrics either consider no transformation at all (e.g. L_1 and L_2 distance metrics) or only the shift of time steps (e.g. cross-correlation). Meanwhile, DTW allows the shifting and deformation to be non-linear. The main drawback of DTW is its computation complexity, which is $O(mn)$ to align two sequences of lengths m and n. While there exist other acceleration or approximation techniques, we accelerate it using GPUs, as described in Section 5.

3.3 Construction of TAC-based Distance Field

By computing the DTW distance from every data point to the feature TAC, the original time-varying volume is transformed into a distance field. This distance field can be visualized using conventional scalar data visualization techniques such as isosurfaces and DVR, as will be described in the next section.

In addition to visualizing the distance field, which is to show the spatial distribution of data points that exhibit a similar temporal trend to the feature TAC, additional information such as when the main feature in the feature TAC moves through space can also be revealed. Using the seismic waves as an example, it is important to know when the peak of the earthquake wave passes through

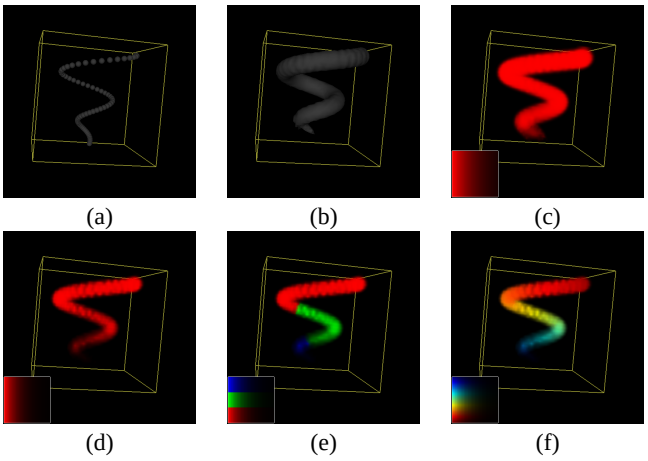


Figure 2: Visualization of the TAC-based distance field of a synthesized data set. This data set is a Gausian kernel moving over time. (a): the centers of the Gaussian kernels. (b): the isosurface. (c) and (d): DVR images where the opacities are decided according to Equation 2 with the drop-off parameter p of 8 and 16, respectively. (e) and (f): DVR images with discrete and continuous, respectively, color textures.

different areas. To highlight this information, along with the feature TAC the user can also indicate where the main feature of interest, the peak of the earthquake wave in this case, appears on the TAC by providing two bounding time steps, which is called *feature time range*. Since DTW will warp the time sequences when matching two TACs, the user-specified feature time range can be warped according to the local data point's TAC. The warped feature time range then indicates when the main feature occurs in the local region. We call the centroid of the warped time range as *feature time step*.

To summarize, in the TAC-based distance field, two attributes are computed for each data point: the DTW distance to the feature TAC and the feature time step when the main feature of interest occurs at that point. The DTW distance records the time-invariant shape dissimilarity between the point's TAC and the feature TAC. The feature time step records the estimated time when the feature passes through that point. By considering both attributes during the visualization process, spatial and temporal distribution of the feature can be revealed.

4 Visualization of TAC-based Distance Field

Once the TAC-based distance field has been constructed, the time-varying feature can be visualized using various visualization techniques. In this section, we use a synthesized data set to illustrate the concepts. This data set was created by moving a 3D Gaussian kernel in different locations over time. The path of the Gaussian kernel is shown in Figure 2 (a). The feature TAC is a 1D Gaussian kernel.

Isosurface To locate voxels that have similar TACs to the feature TAC, isosurfaces of small DTW distances can be extracted. Figure 2 (b) shows an isosurface of the test data set. The isosurface reveals the moving path of the Gaussian kernel, although no information about when the kernel moves through each data point is shown.

Volume Rendering via 1D and 2D Transfer Functions To reveal the temporal information of the feature, we can use DVR to display both the DTW distances and the feature time steps stored in the distance field. This is done using 2D transfer functions to map them to opacity and color.

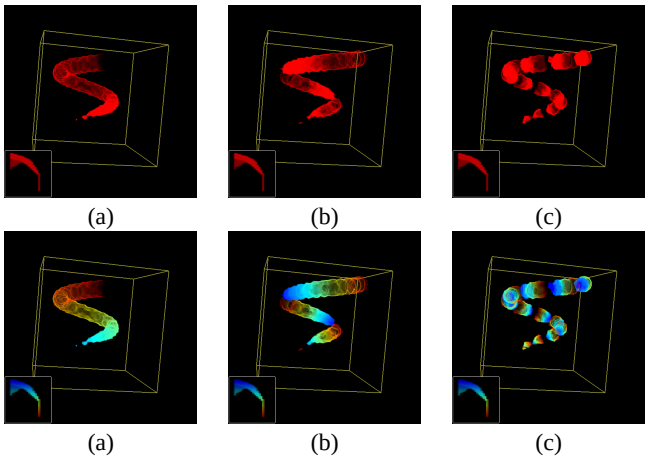


Figure 3: Creating motion trail via DVR. (a), (b) and (c): repeated patterns in different frequencies via discrete color map. (d), (e) and (g): repeated patterns in different frequencies via continuous color map.

When mapping the DTW distance to opacity, our goal is to hide the regions that have too dissimilar TACs to the feature TAC. We use the following equation to calculate the alpha channel for each voxel:

$$\alpha = (1-s)^p \tag{2}$$

where α is the opacity, s is the DTW distance normalized to [0,1] range, and p is a parameter to control the drop-off rate of the opacity as the voxel becomes more dissimilar to the feature TAC. Figure 2 (c) and (d) show images obtained from using different drop-off values. Regions of different sizes are shown to indicate the levels of similarity to the feature.

To map the feature time step to color, there exist several possibilities. One way is to map the feature time steps by a 1D transfer function. Figures 2 (e) and (f) present the DVR images via simple 1D transfer functions that map 'past' to 'red,' 'current' to 'green,' and 'future' to 'blue.' Note that in the figures, the 2D transfer functions are displayed in a small window at the lower left corner. Because the alpha value will drop to a small value near zero as the distance to the feature TAC gets larger, only about a quarter of the color bar is visible.

Another option is to use 2D textures to create different styles of images. The textures can be indexed by using the DTW distance as the row index and the feature time step as the column index. One desired effect is to show the feature's motion trail. To create this effect, the 2D texture can have texels with non-zero opacities arranged as a band covering the distances of interest. The opacities on those texels are controlled by the feature time step to emphasize a certain time range. Figure 3 (a) shows an image generated with this 2D texture. In addition, similar to conventional texture mapping, the texture coordinates can be scaled to created repeated patterns. Figures 3 (b) and (c) show images obtained by using scales of 4 and 16, respectively. Figures 3 (d), (e) and (f) show the images created by using textures with different color channels. Also, effects of contour volumes can be created by using a texture whose opacity channel contains horizontal strips. The examples of contour volumes will be shown in Section 6.

Finally, even though our goal is not to rely on animations to visualize time-varying data, animated effect can be easily incorporated in our framwork. By periodically shifting the column index when accessing the transfer functions, a dynamic trace of the feature can be produced.

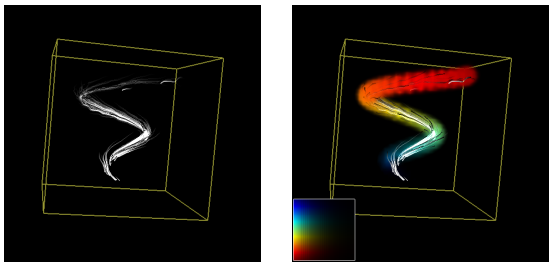


Figure 4: Visualization of the feature motion field for the data set in Figure 2 (a). (a): the streamlines from the feature motion field. (b): combination of both DVR and streamlines.

Motion Visualization over Feature Motion Field In addition to showing the feature in a spatiotemporal manner, we can also create motion traces to depict the feature's movement since such traces provide intuitive cues, as pointed out by Joshi and Rheingans in [7]. While Joshi and Rheingans have used illustration techniques such as speedlines to depict the traces, a feature segmentation stage followed by a feature tracking stage is required. As we mentioned in Section 1, a precise segmentation of time-varying features could be nontrivial. To create motion traces without explicit feature segmentation, our framework converts the TAC-based distance field into a vector field indicating the directions in which the feature may move in the local area. We can then visualize this vector field to infer how the feature may propagate from one data point to another.

We call this derived vector field *Feature Motion Field*. To create the feature motion field, for each voxel $\mathbf{x}$, its motion vector is calculated based on the distances and the feature time steps of the voxel's neighborhood $N(\mathbf{x})$. For a neighboring voxel $\mathbf{y}$, if its feature time step is smaller than or equal to that of $\mathbf{x}$, it is discarded because it indicates that $\mathbf{y}$ is in the past as far as the history of the feature is concerned. Otherwise, the vector $\vec{xy}$ is multiplied by a weight and accumulated to the vector on $\mathbf{x}$. This weight is inversely proportional to the feature distance stored in $\mathbf{y}$ because voxels whose TACs are more similar to the feature TAC should contribute more to characterize the feature's motion. The weight is calculated according to Equation 2, where the input s is replaced by the DTW distance, and the output alpha is now treated as the weight, denoted as $w_s(\mathbf{y})$. After all neighboring voxels around $\mathbf{x}$ have been examined, the final vector's magnitude at $\mathbf{x}$ is weighted by $w_s(\mathbf{x})$ according to Equation 2. Once the feature motion field has been created, we can create the traces using streamlines. Figure 4 shows the derived streamlines for the synthesized Gaussian's kernel, which apparently provide the path of the kernel.

Our feature motion field is similar to the *Feature Flow Field* proposed by Theisel and Seidel [18]. Both techniques rely on conventional streamlines over an intermediate vector field to trace features without explicit feature segmentation. While the concepts are similar, feature flow field can only be derived from a vector field, but our feature motion field can be constructed from any type of time-varying data sets, provided the feature of interest can be described as a TAC.

5 GPU-BASED DTW IMPLEMENTATION

As mentioned in Section 3, the computation of DTW can be expensive. In this section, we describe how GPUs can be used to accelerate the DTW compuation using nVidia's CUDA library [10]. Other than accelerating the warping between the feature TAC and a single input TAC, we also want to increase the throughput by computing the warping of more TACs to the feature TAC in parallel.

In principle, DTW between the feature TAC and all TACs can be accelerated by executing the TACs on multithreads in parallel.

The following properties make DTW suitable to be implemented on GPUs. First, the execution of DTW between each pair of TACs is independent of that of other pairs. Second, according to the recursive Equation 1, elements with the same index will be simultaneously accessed by all threads from all tables; thus this programming flow is suitable for the SIMD architecture of GPUs.

In practice, however, special care should be taken when accessing the tables for dynamic programming due to the high memory access penalty on graphic hardware. Since the warping of feature time ranges needs the resulting table from dynamic programming, the tables for all TACs must be maintained when executing DTW, requiring an enormous memory space that can be only stored in the device memory (called *global memory* in CUDA terminology). Accessing the global memory, however, is slow since there is no cache. If these tables are unfolded and cascaded together to form a long vector, one instruction to access the elements of the same index will issue multiple memory requests to the global memory, thus causing high memory access penalty. In our experiment, such GPU-based implementation was even slower than the CPU-based implementation on a quad-core system, in spite that GPUs can simultaneously execute DTW for more TACs.

To address the issue, the tables for all TACs are interleaved by cascading the elements with the same index into a contiguous memory address. Therefore, the instruction to access the elements of the same index will issue only one memory request. This modification can made the GPU-based implementation at least ten times faster than the CPU-based implementation in our experiment. For example, for a data set of $150 \times 62 \times 62$ voxels with 200 time steps, the CPU-based implementation took 135 seconds on a workstation with two Dual Core AMD Opteron processors and 8GB system memory, while this optimized GPU-based implementation took only 10 seocnds on an nVidia GeForce 8800GTX card.

6 Case Studies

To demonstrate the utility of our framework, we applied it to data generated from two different applications. One data set is *TeraShake 2.1* [11], the benchmark for the IEEE Visualization Design Contest 2006. The other data set is the benchmark for the IEEE Visualization Design Contest 2008, generated by a simulation that models ionization front instability [19].

6.1 Earthquake Wave Simulation

The data set TeraShake 2.1 records a simulated earthquake in 250 seconds on the Southern San Andreas Fault. During the first 60 seconds of simulation, a magnitude 7.7 earthquake began from South of Palm Springs toward Northwest on the San Andreas Fault, then the fault rupture stopped while the earthquake waves kept propagating. The original data set is a vector field at a resolution of $750 \times 375 \times 100$ voxels with 227 time steps. In our test, the original data set was down-sampled, to reduce the overall storage, by approximately a factor of four in all three dimensions yielding a data set of $188 \times 94 \times 25$ voxels with 227 time steps.

The first property in which we were interested is the propagation of the seismic energy. We used the magnitude of seismic wave to create the TACs. Figure 5 (a) presents the mean TACs from three clusters, while the chosen feature TAC is plotted in blue. It shows that all three mean TACs contain a peak in different locations, followed by a tailing signal of different lengths. This difference suggests that using DTW to calculate the dissimilarity will obtain a smaller distance than L_1 distance, L_2 distance, or the cross-correlation metrics.

Figure 5 (b) shows the joint histogram of the feature time step and DTW distance. The DTW distance and the feature time step are represented as x and y coordinates, respectively. The occurrence of each bin is normalized to [0, 1] and then mapped to color from 'blue' to 'red.' The bins with zero occurrence are not drawn. In the joint histogram, after the 100th time step, the horizontal distance between the y-axis and the first non-empty bin is increasing as the y coordinate increases, which means the number of voxels that contain wave magnitudes closer to the maximal magnitude of the feature TAC is decreasing in time, indicating that the strength of the earthquake is reducing after the 100th time step. This is consistent with the simulation setting that the fault rupture was terminated at the 60th second or equivalently the 55th time step.

Figure 5 (c) presents the DVR image of the TAC-based distance field. Here the transition of color indicates the propagation paths of the earthquake energy. For instance, the path where the color varies from red to yellow is along the structure of the basin plotted as the isosurfaces. The wavefronts of the reflected waves are also depicted. Figure 5 (d) provides the streamlines computed from the feature motion field. Compared to Figure 5 (c), those streamlines show clearer propagation paths of the earthquake waves. It suggests that streamlines allow a better inference about the propagation paths than DVR images. On the other hand, volume rendering depicts better the temporal transition of the energy using colors. To combine their advantages, Figure 5 (e) shows both the DVR result and the streamlines.

The types of the seismic waves are another interesting phenomena. Our focus was to reveal when and where the types of waves changed from body waves (S- or P- waves) to surface waves (L- or R- waves) and vice versa. Because the types of waves are determined by the vector's orientation, the original vector field was transformed into a scalar field called *orientation field* as a measurement of the orientation disparity between consecutive vectors. In the orientation field, the scalar stored in each voxel $\mathbf{x}$ at time step t is the product of the magnitudes of the two vectors at time steps t and $t+1$ at the voxel $\mathbf{x}$ multiplied by an indicator constant. This indicator constant is $+1$ if the disparity is larger than 45 degree or -1 otherwise.

Figure 6 (a) presents the feature TAC. Since a dramatic change of the orientation will cause a negative value in the orientation field, the feature TAC contains a negative peak. Figure 6 (b) presents the joint histogram of the TAC-based distance field. In this joint histogram, voxels with distances smaller than 0.1 are mainly distributed before the 70th time step. This distribution indicates that the change of wave types occurred at the first one-third of the simulation.

Figure 6 (c) shows the DVR image. Compared to Figure 5 (c), even though a 2D transfer function with a larger support of non-zero opacity was used, Figure 6 (c) still appears more transparent than Figure 5 (c), which makes it difficult to find exactly where the wave types changed. The streamlines from the feature motion field are shown in Figure 6 (d). Compared to volume rendering in Figure 6 (c), streamlines suggest a clearer view of the orientation field and indicate where the wave types changed. Figure 6 (e) shows two sets of streamlines computed from the feature motion fields: one was generated by the magnitude and the other by the orientation. As shown in Figure 6 (e), both sets are propagating along a similar direction, although the streamlines derived from the orientation field are less correlated with the direction than the streamlines derived from the magnitude field.

6.2 Ionization Front Instability

Our second case study is to visualize the turbulence of ionization front instability. This data set was used for the 2008 IEEE Visualization Design Contest, whose aim is to reveal the influence of the I-front instabilities during the formation of the structure in early universe [19].

The data set is a multifield time-varying volume. Each voxel records 10 fields and a 3D velocity vector. This data set contains 200 time steps of $600 \times 248 \times 248$ voxels. We downsampled the data set by a factor of four in all three dimensions to reduce the

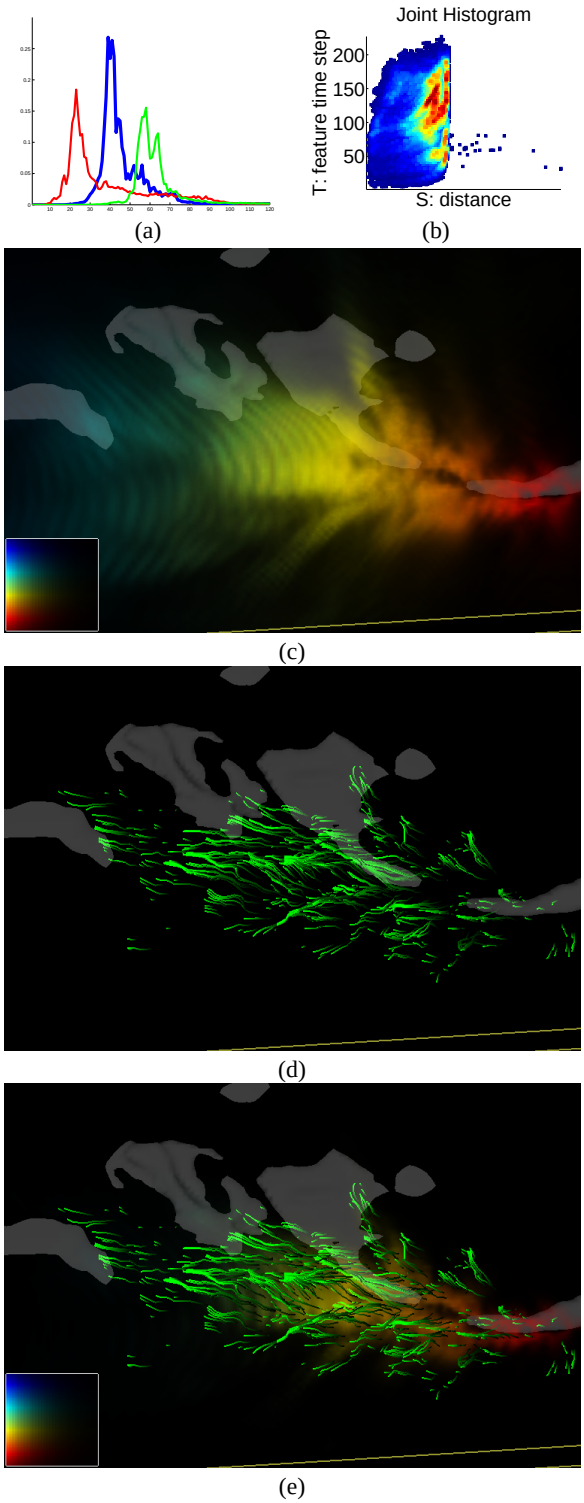


Figure 5: Visualization of the magnitude field of the TeraShake 2.1 data set. (a): the mean TACs from three clusters; the selected feature TAC is plotted in blue. (b): the joint histogram. (c): a DVR image. (d): streamlines from the feature motion field. (e): combination of the DVR image in (c) and the streamlines in (d).

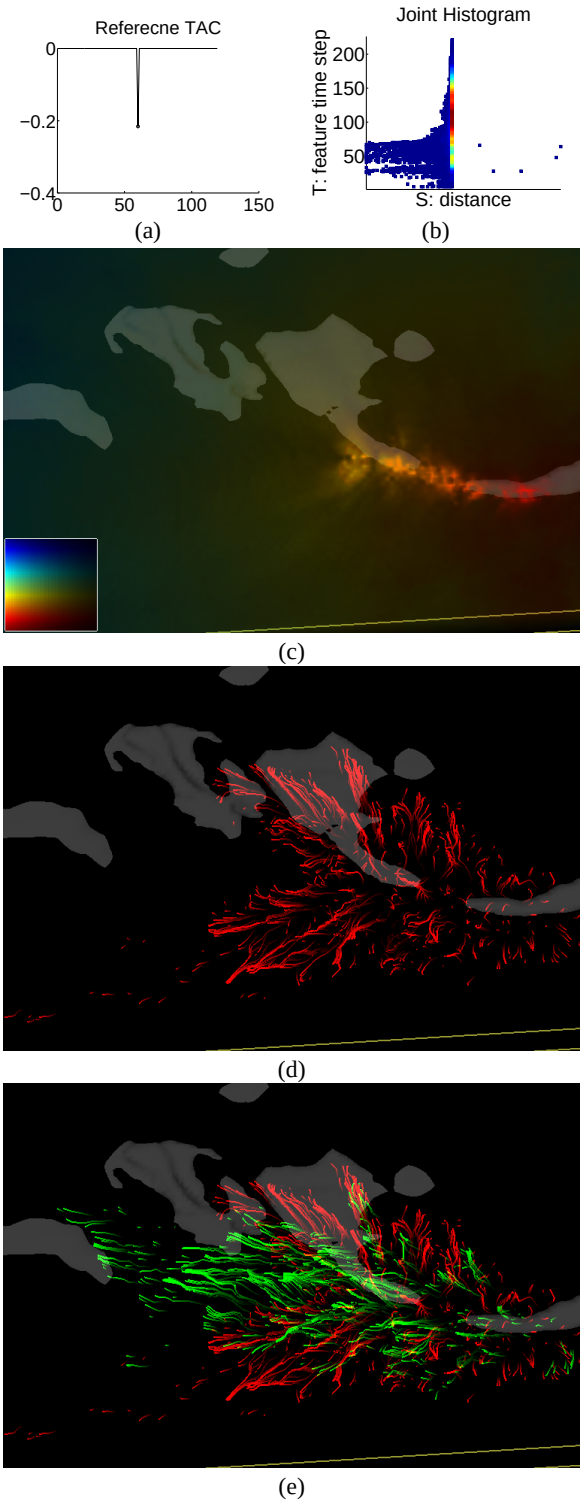
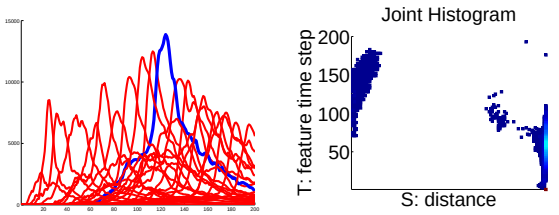


Figure 6: Visualization of the orientation field of the TeraShake 2.1 data set. (a): the feature TAC. (b): the joint histogram. (c): a DVR image. (d): streamlines from the feature motion field. (e): combination of the streamlines from the magnitude field (green) and the orientation field (red).

storage. Because the scale of the turbulence is related to the scale of the curl field, we converted the velocity field into the curl field, from which the TACs were created.

Figure 7 (a) shows the feature TAC in blue. This feature TAC



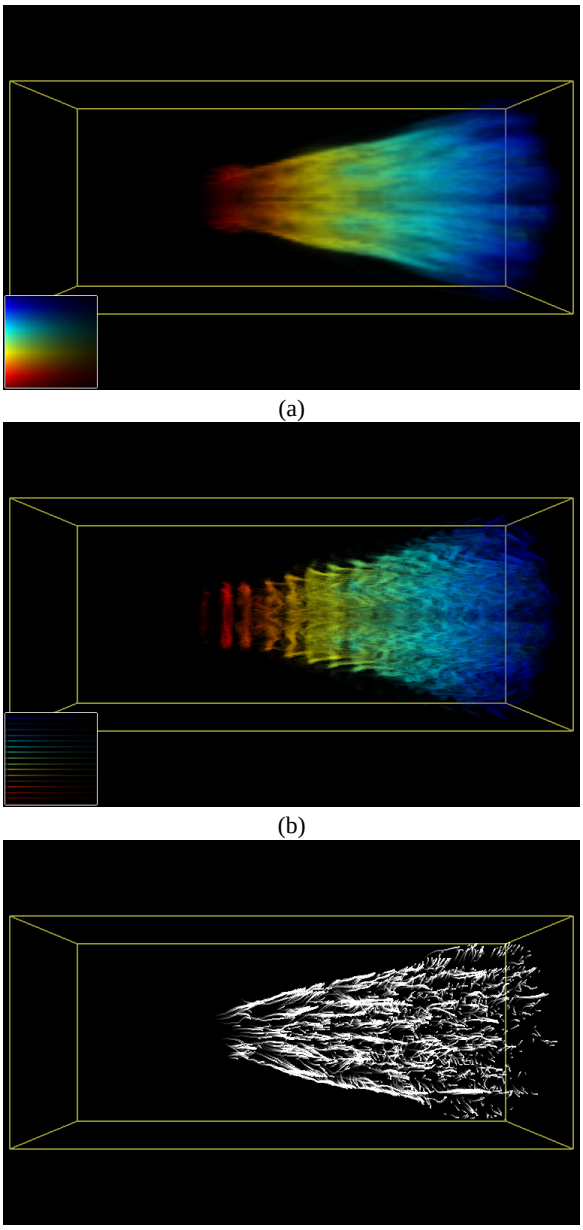
(a) (b)

Figure 7: The feature TAC for the ionization front instability simulation. (a): the mean TACs of all clusters (red) and the feature TAC (blue). (b): the joint histogram of the respective TAC-based distance field.

was extracted via K-mean clustering. The mean TACs from the clusters are plotted in red. It can be seen that most mean TACs show an impulse at different time steps and with different durations. Figure 7 (b) shows the joint histogram of the corresponding TAC-based distance field. In the joint histogram, after the 70-th time step, the number of voxels with distances smaller than 0.25 is increasing over time. This indicates that the turbulence mainly occurred after the 70th time step, and its structure was expanding over time.

We are mainly interested in the spatiotemporal structure of the turbulence. Figure 8 (a) presents a DVR image. The colors of voxels from left ($-x$) to right ($+x$) vary from red to blue, indicating the propagation path of the turbulence. This is consistent with the simulation setting that the I-front propagated along the x axis. While Figure 8 (a) provides an overview of the propagation, it cannot reveal the detail of the turbulence structure. Thus a texture with horizontal strips was applied to create contour volumes. The result is shown in Figure 8 (b). The contours show that the structure of the turbulence was distorting as the I-front propagated. Figure 8 (c) shows the streamlines generated from the corresponding feature motion field. It can be seen that there is no streamline near the left side of the bounding box, which is the source of the I-front. This lack of streamlines suggests that there was no turbulence at the beginning of the simulation. It is noteworthy that in Figure 8 (c), the streamlines in the middle part are longer than the streamlines in the right side, which suggests that the propagation of the turbulence was obstructed near the end of the simulation.

The symmetry of the turbulent structure is also of interest. Figure 9 contains subfigures rendered from a viewing direction parallel to the $x-$ axis. The contours in Figure 9 (a) indicate that the major structure of the turbulence is symmetric about the x axis. When the image is enlarged, however, minor non-symmetry is apparent as shown in Figure 9 (b). Figure 9 (c) shows the streamlines computed from the feature motion field. From the distribution, it can be seen that more non-symmetry is revealed. Figure 9 (d) shows the combination of the streamlines and the DVR result.



(a) (b) (c)

Figure 8: Visualization of the ionization front instability simulation. (a): a DVR image with continuous color map. (b): a DVR image with contour volumes. (c): streamlines from the feature motion field.

7 Conclusion

In this paper, we present a new framework for visualizing time-varying data sets. We are interested in those features that can be modeled as time series patterns. Since the shapes of the patterns could be shifted, stretched, or compressed over time, we use DTW as the time-invariant distance metric. By converting the original sequence of volumes into a distance field to a specified feature, we can visualize the spatiotemporal behavior of the feature via volume rendering. To emphasize the propagation of the feature, we apply flow visualization techniques over an intermediate feature motion field derived from the distance field to create the motion traces of the feature.

The benefit of our framework is multifold. By using DTW, The distance between feature descriptors represented as TACs can be computed invariantly under time warp. DTW also provides a scheme to enable precise temporal inference to the feature. An overview of the time-varying feature can be displayed in a single image rather than in animation, and different visualization styles can be created.

Despite the advantages, there exist several limitations in our current framework. First, the TAC-based distance field cannot be created until the entire volume of time sequences becomes available. Thus our method could be inadequate for online visualization. Second, our algorithm prefers data that exhibits spatial coherence. For features with little spatial overlap in adjacent time steps, discontinuous patterns may appear in the visualization. Third, streamlines computed from the feature motion field might not reveal whether the features are splitting or merging. Finally, even though DTW

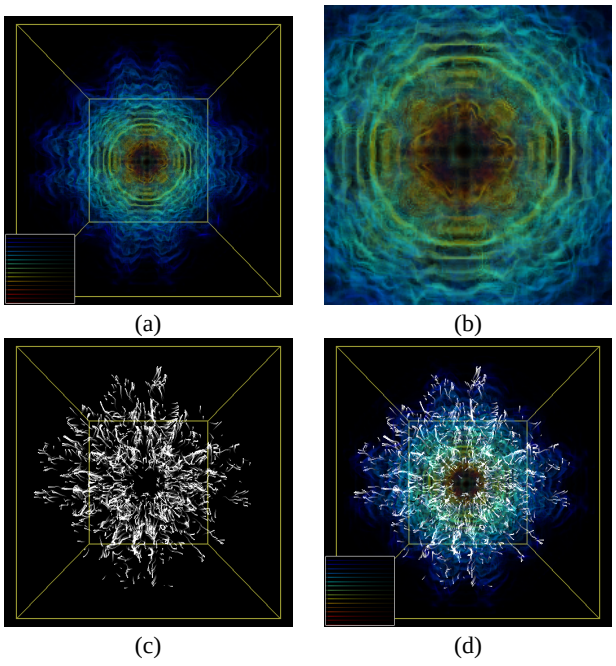


Figure 9: Visualization of the ionization front instability simulation. Compared to Figure 8, the subfigures here were rendered along a viewing direction parallel to the x axis, thus providing better cue about the symmetricity of the turbulence structure. (a): a DVR image . (b): the enlarged image of subfigure (a). (c): the streamlines from the feature motion field. (d): combination of the DVR image in (a) and the streamlines in (c).

metric is invariant under the shift of time, it does not encounter the change of scales in the TACs.

In the future, in order to define more types of features, the use of TACs in multi-dimensions rather than 1D scalars should be studied. In the applications where TAC patterns with the same shape but in different scales are considered equivalent, modified DTW such as Derivative Dynamic Time Warping [8] will be needed. Moreover, currently only the DTW distances and the feature time steps are considered. The feature time ranges can also be embedded in the rendering process to create more effective visualization. Creating a more robust feature motion field should be studied too in order to trace features that do not spatially overlap. Yet other direction is to transform the distance field into the scale space, where a feature can be detected across different scales. One example of such techniques is the SIFT feature detector [9] used in computer vision applications.

REFERENCES

[1] D. Bauer and R. Peikert. Vortex tracking in scale-space. In *VisSym '02: Proceedings of the Joint Eurographics-IEEE TCVG Symposium on Visualization 2002*, pages 140–147, 2002.

[2] J. Caban, A. Joshi, and P. Rheingans. Texture-based feature tracking for effective time-varying data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1472–1479, 2007.

[3] Z. Fang, T. Möller, G. Hamarneh, and A. Celler. Visualization and exploration of time-varying medical image data sets. In *GI '07: Proceedings of Graphics Interface 2007*, pages 281–288, 2007.

[4] H. Guo, R. Renaut, K. Chen, and E. Reiman. Clustering huge data sets for parametric pet imaging. *BioSystems*, 71(1–2):81–92, 2003.

[5] G. Ji and H.-W. Shen. Feature tracking using earth mover's distance and global optimization. In *Proceedings of Pacific Graphics 2006*, 2006.

[6] G. Ji, H.-W. Shen, and R. Wenger. Volume tracking using higher dimensional isosurfacing. In *VIS '03: Proceedings of the IEEE Visualization 2003*, pages 209–216, 2003.

[7] A. Joshi and P. Rheingans. Illustration-inspired techniques for visualizing time-varying data. In *VIS '05: Proceedings of the IEEE Visualization 2005*, pages 679–686, 2005.

[8] E. J. Keogh and M. J. Pazzani. Derivative dynamic time warping. In *Proceedings of the First SIAM International Conference on Data Mining*, 2001.

[9] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[10] NVIDIA Corporation. *NVIDIA CUDA Compute Unified Device Architecture Programming Guide 1.1*, 2007.

[11] K. B. Olsen, S. M. Day, J. B. Minster, Y. Cui, A. Chourasia, M. Faerman, R. Moore, P. Maechling, and T. Jordan. Strong shaking in los angeles expected from southern san andreas earthquake. *GEOPHYSICAL RESEARCH LETTERS*, 33:L07305, 2006.

[12] F. H. Post, F. J. Post, T. V. Walsum, and D. Silver. Iconic techniques for feature visualization. In *VIS '95: Proceedings of the IEEE Visualization '95*, pages 288–295, 1995.

[13] F. Reinders, F. H. Post, and H. J. Spoelder. Attribute-based feature tracking. In E. Gröller, H. Löffelmann, and W. Ribarsky, editors, *VisSym '99: Proceedings of the Joint Eurographics-IEEE TCVG Symposium On Visualization '99*, pages 63–72, 1999.

[14] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[15] D. Silver and X. Wang. Volume tracking. In *VIS '96: Proceedings of the IEEE Visualization '96*, pages 157–164, 1996.

[16] D. Silver and X. Wang. Tracking scalar features in unstructured datasets. In *VIS '98: Proceedings of the IEEE Visualization '98*, pages 79–86, 1998.

[17] N. A. Svakhine, Y. Jang, D. S. Ebert, and K. P. Gaither. Illustration and photography inspired visualization of flows and volumes. In *VIS '05: Proceedings of the IEEE Visualization 2005*, pages 687–694, 2005.

[18] H. Theisel and H.-P. Seidel. Feature flow fields. In *VisSym '03: Proceedings of the Joint Eurographics-IEEE TCVG Symposium on Visualization 2003*, pages 141–148, 2003.

[19] D. Whalen and M. L. Norman. Competition data set and description. 2008 IEEE Visualization Design Contest, 2008. http://vis.computer.org/VisWeek2008/vis/contests.html.

[20] K.-P. Wong, D. Feng, S. Meikle, and M. Fulham. Segmentation of dynamic pet images using cluster analysis. *IEEE Transactions on Nuclear Science*, 49(1):200–207, 2002.

[21] J. Woodring and H.-W. Shen. Chronovolumes: A direct rendering technique for visualizing time-varying data. In *Proceedings of the Third International Workshop on Volume Graphics 2003*, pages 27–34, 2003.

[22] J. Woodring, C. Wang, and H.-W. Shen. High dimensional direct rendering of time-varying volumetric data. In *VIS '03: Proceedings of the IEEE Visualization 2003*, pages 417–424, 2003.