Salient Time Steps Selection from Large Scale Time-Varying Data Sets with Dynamic Time Warping

Xin Tong Teng-Yok Lee Han-Wei Shen

The Ohio State University *

ABSTRACT

Empowered by rapid advance of high performance computer architectures and software, it is now possible for scientists to perform high resolution simulations with unprecedented accuracy. Nowadays, the total size of data from a large-scale simulation can easily exceed hundreds of terabytes or even petabytes, distributed over a large number of time steps. The sheer size of data makes it difficult to perform post analysis and visualization after the computation is completed. Frequently, large amounts of valuable data produced from simulations are discarded, or left in disk unanalyzed. In this paper, we present a novel technique that can retrieve the most salient time steps, or key time steps, from large scale time-varying data sets. To achieve this goal, we develop a new time warping technique with an efficient dynamic programming scheme to map the whole sequence into an arbitrary number of time steps specified by the user. A novel contribution of our dynamic programming scheme is that the mapping between the whole time sequence and the key time steps is globally optimal, and hence the information loss is minimum. We propose a high performance algorithm to solve the dynamic programming problem that makes the selection of key times run in real time. Based on the technique, we create a visualization system that allows the user to browse time varying data at arbitrary levels of temporal detail. Because of the low computational complexity of this algorithm, the tool can help the user explore time varying data interactively and hierarchically. We demonstrate the utility of our algorithm by showing results from different time-varying data sets.

1 INTRODUCTION

As the speed of processors and degree of parallelism continue to increase, the high cost of storage and slow speed of data movement have now become the major limiting factors for applications to take full advantage of the computation power in large scale parallel machines. For scientific applications such as climate modeling, nuclear reactor simulation, and combustion engine design, the overwhelming amount of data generated by time-varying simulations forces the scientists to cut down the amount of data that can be stored to disk. Since it is not possible to analyze results from all time steps, scientists often can only look at a small fraction of data at some pre-selected time interval decided with simple heuristics. When the phenomena modeled by the simulations have complex patterns and occur at unknown frequency, the question of what time steps to use for analysis and visualization is still illusive.

Previously researchers have proposed various methods to select salient time steps from time-varying data sets. One approach is based on user input. By providing an overview of the time-varying data in a 2D layout, the user can judge what time steps are more important and hence should be further investigated. Examples of this approach include the spreadsheet-like layout for time series data [20], storyboard-like layout in the form of images [13], or the time histogram method that concatenates data distributions over time [2, 1]. For certain applications such as data compression or in situ analysis, nevertheless, it will be challenging to have direct intervention from the users so automatic approaches are more preferred. For this purpose, there exist several algorithms that can find representative time steps. One method is to group similar time steps using greedy approaches, and then choose one time step from each group. Such an approach is taken by Akiba *et al.* [1] to group time steps based on data distribution. Also, a time step can be deemed salient if it changes drastically from the previous time steps. Based on this concept, Wang *et al.* measure the mutual information in local time steps to see when the distribution has a larger amount of changes. [17]. Those approaches, however, only consider the data in a local time range. Given a constraint in the total number of time step to select, the selected time steps can be suboptimal if the global information over the entire time sequence is not considered.

To allow scientists to focus on the most salient data for analysis, in this paper, we present a novel technique for identifying key time steps from a time-varying data set using a global optimization scheme. A key time step is a time step that can best represent the data in its surrounding time steps. Given a user-desired number of key time steps and a distance metric to compare data between two time steps, time steps that can minimize the total cost of representing the whole data sequence are chosen as the key time steps. Our technique is inspired by *D*ynamic Time Warping (DTW), a technique that non-linearly warps one time series to another at a minimum cost. Instead of mapping two different time series, as in most of the DTW applications, in our technique we treat the input time-varying data as one time series and warp it to a subset of its own time steps. We design an efficient computation algorithm based on dynamic programming that can rapidly identify the time steps with the minimum cost under nonlinear time warping. We present a data browsing tool that can allow the user to interactively visualize and analyze time varying data with the key time steps selected by our algorithm. The key time steps selection process can be done very efficiently and hence allows the user to select any number of key time steps in real-time. When the user detects an interesting time interval by visualizing the current key time steps, they can request additional time steps in the time interval, invoking our key time step selection algorithm recursively.

We apply our key time step selection algorithm to a cloud-resolving simulation using the Earth Mover's Distance [15] as the dissimilarity measure. From the visualization and analysis of the data in the key time steps, we can observe salient moments of the Madden-Julian Oscillation, a long term weather event. Another application of our algorithm discussed in this paper is to identify key time steps for time varying isosurfaces. Based on the isosurface dissimilarity map [3], our algorithm allows the user to obtain insight into the temporal evolution of isosurfaces.

The structure of the paper is organized as follows. First, we review the related work on key time step selection and temporal classification of time-varying data in Section 2 Then we present the general idea of mapping in DTW, and the problem of mapping for key time steps selection in Section 3. An efficient dynamic pro-

*e-mail: {tong,leeten,hwshen}@cse.ohio-state.edu

IEEE Symposium on Large Data Analysis and Visualization 2012
October 14 - 15, Seattle, Washington, USA
978-1-4673-4733-4/12/$31.00 ©2012 IEEE

gramming algorithm to solve the mapping problem is described in Section 4. Results from case studies using different data sets are discussed in Section 5. A time-varying data browser based on our algorithm is described in Section 6. Section 7 discusses the limitations of our approach and potential future works. Finally, we conclude the paper in Section 8.

2 Related Works

Researchers have focused on various aspects of time-varying data visualization in the past two decades. Examples are feature tracking, compression, high-dimensional rendering, and interaction. Because the goal of this paper is to detect representative time steps for visualization and exploration, here we mainly focus on works related to effective browsing of time-varying data.

One strategy to overview time-varying data is to depict the information from all time steps simultaneously. As histograms are frequently used to display data distributions, Akiba *et al.* designed a visualization interface based on *Time Histogram* [1]. Time Histogram is a 2D image that concatenates data distributions over time, allowing the user to design effective transfer function across multiple time steps. Later Time Histogram was combined with parallel coordinates for multivariate volume exploration [2]. Besides, other statistics over time can be utilized too. One example is the importance curves presented by Wang *et al.* [17]. For each spatial block, its importance curve is computed from the mutual information between adjacent time steps over time. As mutual information represents the similarity between local time steps, importance curves can indicate when the distribution within a data block has changed.

In addition to considering data distributions, direct visualization of value changes over time can be beneficial too. Given a fixed spatial location, the data values over time form a time series, or called Time Activity Curve (TAC) in medical applications [19, 6]. In the user interface designed by Fang *et al.*, data in the spatial domain can be classified based on their TAC patterns, and the region of interest can be detected based on user-specified TACs [4] . Woodring and Shen [20] used k-means clustering to detect the representative TACs from all data points, and designed a spreadsheet-like interface to overview all representative TAC and its distribution over the spatial domain. Later Woodring and Shen use the TAC-based representation to detect the evolution of features, and create a transfer function that can be automatically adapted to capture the change of data ranges[21]. To track the propagation of feature, Lee and Shen presented an idea called TAC-based distance field [11]. Given a user-specified feature, their approach is to first model the feature as a TAC, and then estimate when and where the feature TAC appears in all data TACs. Based on the estimated locations and time steps, the propagation of the feature can be depicted in a 2D plot.

Because of the limited screen space, it is also beneficial to only display a subset of time steps that can represent the change of importance within the data. Lu and Shen presented a storyboard-like interface [13] where the key time steps are distributed on 2D screen space based on similarity measures of the data. By linking these key time steps following the order of time, the user can see how the data evolve. Lee and Shen modeled the change of TAC pattern as transition of states, thus reducing the TAC at each location into a state machine that has fewer states [10]. The concept of state-transition graph is also used by Wang *et al.* [16] and the *TransGraph* presented by Gu and Wang [5]. Along this direction, our goal is to select salient time steps that can be the most representative for the whole sequence. We also design effective user interfaces to allow effective browsing of time-varying data.

3 Time Sequence Alignment

The goal of this work is to identify the best K time steps from a time-varying data set of N time steps, where K is specified by the user. The selected time steps should be the most representative among all the possible choices, i.e., they should maximize the information conveyed by the original data. Since K is smaller than N, to estimate the cost of choosing the K key time steps we need to first align the original time-varying data to the selected subset. For this, we can consider the entire data set as a time series, where each time point represents the volume data at one time step. The concept of time sequence alignment appears in the Dynamic Time Warping(DTW) algorithm. DTW algorithm generates an optimal alignment between two time series, and measures the cost of the alignment based on the similarity between the time points in two series. Inspired by DTW, we apply a similar method to measure the cost of key time steps. In our case, the time series is mapped to a subsequence of its own. Our goal is to find the best K time steps that give the minimum cost determined by DTW. In the following, we describe our approach in detail.

3.1 Alignment in Dynamic Time Warping

Given two time sequences X and Y, to measure the similarity between the two sequences, a non-linear mapping can be done to align points considering their similarity. An example of such a mapping is shown in Figure 1.

To represent the mapping, let an alignment between two sequences $X = (x_1, x_2, ..., x_N)$ and $Y = (y_1, y_2, ..., y_M)$ be $P = \{p_1, p_2, ..., p_L\} = \{(x_{n_1}, y_{m_1}), (x_{n_2}, y_{m_2}), ..., (x_{n_L}, y_{m_L})\}$. Each (x_{n_i}, y_{m_i}) is a bi-directional mapping between x_{n_i} in sequence X and y_{m_i} in sequence Y, and L is the total number of pairs between X and Y. Because one element in a sequence can map to multiple elements in the other sequence and vice versa, and P lists all the element-element alignment pairs, the number of pairs L in P, is equal to or greater than both M and N, i.e., $L \geq M$ and $L \geq N$. Figure 1 shows the alignment pairs in green lines. Because of the non-linear nature, it can be seen that an element on one sequence can map onto more than one element on the other sequence. For example, x_1 maps to both y_1 and y_2, $x_{n_1} = x_{n_2} = x_1$.

To measure the cost of aligning two sequences X and Y, a dissimilarity function between two data points x_{n_i} and y_{m_i}, $D(x_{n_i}, y_{m_i})$, is defined to measure the difference between the two elements. Then, the *overall mapping cost* of the alignment is defined as [14]:

$$C = \sum_{i=1}^{L} D(x_{n_i}, y_{m_i}) \qquad (1)$$

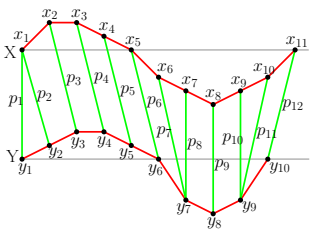


Figure 1: Alignment between two sequences X and Y

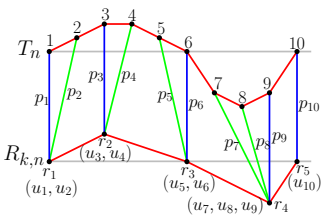
DTW cost is defined as the overall minimum cost among the set of all possible mappings $\mathbb{P}$:

$$C^* = \min_{P \in \mathbb{P}}(C) = \min_{P \in \mathbb{P}} \left(\sum_{i=1}^{L} D(x_{n_i}, y_{m_i}) \right) \qquad (2)$$

This lowest overall mapping cost defines an *optimal alignment* P^*, which is the dissimilarity between these two sequences. DTW uses a dynamic programming algorithm to generate the optimal alignment efficiently.

3.2 Mapping in Key Time Steps Selection

With the basic concept of DTW explained, in this section we discuss how key time steps can be selected from a full time series. To simplify the notations, hereafter we only list the time step indices rather than the actual data to represent a time sequence.



(a)

p_i	(i,u_i)	value
p_1	$(1,u_1)$	$(1,r_1)$
p_2	$(2,u_2)$	$(2,r_1)$
p_3	$(3,u_3)$	$(3,r_2)$
p_4	$(4,u_4)$	$(4,r_2)$
p_5	$(5,u_5)$	$(5,r_3)$
p_6	$(6,u_6)$	$(6,r_3)$
p_7	$(7,u_7)$	$(7,r_4)$
p_8	$(8,u_8)$	$(8,r_4)$
p_9	$(9,u_9)$	$(9,r_4)$
p_{10}	$(10,u_{10})$	$(10,r_5)$

(b)

Figure 2: (a): Mapping between full sequence T_n and key sequence $R_{k,n}$ (b): DTW mapping between the two sequences in (a)

Given a sequence of n time steps $T_n = \{1,2,...,n-1,n\}$, called *full sequence*, we can choose k time steps $R_{k,n} = \{r_1, r_2, ..., r_{k-1}, r_k\}$ from the full sequence, where $1 \leq r_1 < r_2 < ... < r_{k-1} < r_k \leq n$. There are many possible selection of $R_{k,n}$. We want the one $R_{k,n}$ that is the most similar to the full sequence T_n. To achieve this, we will need a similarity measure for the two sequences.

If we apply DTW to T_n and $R_{k,n}$, the overall mapping cost would tell us how similar they are. To evaluate the cost, we start with constructing an alignment between the full sequence and the selected sequence, as shown in Figure 2(a).

The mapping is defined by $P = \{p_1, p_2, ..., p_n\} = \{(1,u_1),(2,u_2),...,(i,u_i),...,(n,u_n)\}$. Each (i,u_i) means time step i maps to time step u_i, where $u_i \in R_{k,n}$. In the example shown in Figure 2(a), All $n = 12$ mappings of P between the two sequences T_n and $R_{k,n}$ are listed in Figure 2(b). Since in our case each time step i in T_n has only a unique key time step r_i in $R_{k,n}$ to map to, we can simplify the more general bi-directional mapping mentioned in section 3.1 by a uni-directional mapping function $u_i = f(i)$, which is a special case of bi-directional mapping. On the other hand, a time step r_i in $R_{k,n}$ still may have multiple time steps in T_n to be mapped onto. From the example, it can be seen that time step 1 and time step 2 are mapped to r_1, so in the mapping $u_1 = u_2 = r_1$.

Let the dissimilarity between time step i and time step u_i be $D(i,u_i)$. The *DTW cost* of mapping between T_n and $R_{k,n}$, according to Equation 2, is defined as:

$$C_{k,n} = \min_{P \in \mathbb{P}} \left(\sum_{i=1}^{n} D(i,u_i) \right) \tag{3}$$

The specific P that gives the lowest mapping cost $C_{k,n}$ is the optimal mapping, denoted as P^*, for a given $R_{k,n}$.

Given a n-step time series, there are many different ways to choose k time steps, i.e. many possible $R_{k,n}$. If we simply exhaust all the possible $R_{k,n}$, and choose the one with the lowest DTW cost as the *optimal key sequence* or *optimal key time steps*, we have to evaluate:

$$R^*_{k,n} = \arg\min_{R_{k,n}} C_{k,n} = \arg\min_{R_{k,n}} \left(\min_{P \in \mathbb{P}} \left(\sum_{i=1}^{n} D(i,u_i) \right) \right) \tag{4}$$

Clearly, the amount of computation to find $R^*_{k,n}$ is going to be very large. To avoid the huge computation, in the next section we present an efficient approach to select the optimal key sequence.

4 Algorithm

To allow efficient selection of key time steps based on Equation 4, we need to place three constraints. First, each time step in the original time sequence maps to one and only one key time step. Also, since the key time steps in $R_{k,n}$ is to represent the full time sequence T_n, it does not make sense if the key time step does not represent itself. Therefore, our second constraint is that any key time step r_i in the selected sequence should be mapped to itself in the original time sequence, i.e., $u_{r_i} = r_i$, as shown in the blue lines in Figure 2(a). The third constraint is that the last time step in the original time sequence is always used as a key time step, i.e., $r_k = n$. In fact, if this is undesired, we can easily amend it by adding an artificial time step in the end of the full sequence that is very different from all the other time steps in T_n. We can select one more key time steps from the total number of $n+1$ time steps, and discard this artificial step after $R^*_{k+1,n+1}$ is generated.

Given k key time steps, the full sequence is divided into k segments, as shown in Figure 3. Each segment $S_{i,j}$ is bounded by two key time steps, time step i and time step j, i.e. $S_{i,j} = \{i, i+1, ..., j-1, j\}$. Note that the first segment S_{1,r_1} always starts from the first time step and hence bounded only by one key time step, r_1.

Each segment $S_{i,j}$ represents a mapping to the key time steps i and j, and is associated with a cost, referred to as the *segment cost* and denoted as $\| S_{i,j} \|$. This cost is determined by DTW, as will be explained later. The overall cost of the mapping from the full sequence to the key time steps is the sum of the costs from all k segments, denoted as $C_{k,n}$, and can be computed as:

$$C_{k,n} = \| S_{1,r_1} \| + \sum_{i=1}^{k-1} \| S_{r_i,r_{i+1}} \| \tag{5}$$

Equation 5 in fact solves the same problem as Equation 3, except that the mappings in each segment can be solved independently because of the constraint that each key time step maps to itself. This constraint forces each time step to map to itself because DTW prohibits the mapping to contain crossing between the segments, which allows us to solve the DTW cost of each segment independently.

The result of the optimal key time steps selection is $R^*_{k,n}$, which gives the lowest $C_{k,n}$ value, that is:

$$R^*_{k,n} = \arg\min_{R_{k,n}} \left(\| S_{1,r_1} \| + \sum_{i=1}^{k-1} \| S_{r_i,r_{i+1}} \| \right) \tag{6}$$

Similarly, Equation 6 gives the same result as Equation 4 because all possible selections of key time steps $R_{k,n}$ are considered, and the optimal mapping is used within each segment.

The simplest but a brute force way to solve the above equation is to exam all the $\binom{n}{k}$ possible combinations of $R_{k,n}$, and then pick the one with the lowest $C_{k,n}$ value as the optimal key time steps. However, the number of trials, $\binom{n}{k}$, is going to be very large when n and k are large, which makes it impractical.

Actually, the DTW cost computations of different selection of $R_{k,n}$ are not independent. If two key sequences share a subsequence of key time steps, the DTW mapping within this subsequence is the same, and hence need not be computed more than once. Dynamic programming can be used to avoid this redundant computation, because it stores each step's result for later use. In the next section, we describe how dynamic programming can be used to solve the problem.

4.1 Dynamic Programming Selection

Dynamic programming divides a problem into sub-problems, and solve the sub-problem in the same way as the original problem until a stop condition is reached.

Suppose the mapping costs of all possible segments $S_{i,j}$ are known. We want to find the selection $R_{k,n}$ for the lowest $C_{k,n}$ value. Liu et al. proposed a dynamic programming to select key frames from video by maximizing the energy function [12]. Their method

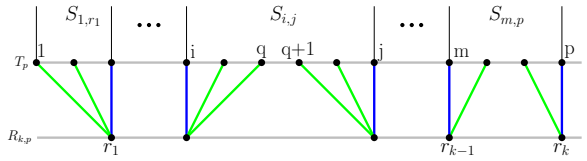


Figure 3: Segments in the mapping between T_p and $R_{k,p}$

could be adapted to help us figure out the key time steps and minimize the DTW cost.

Let $T_p = \{1,2,...,p\}$ be the first p time steps. Let $R_{k,p} = \{r_1, r_2, ..., r_{k-1}, r_k\}$ be the selection of the k key time steps from T_p, and the corresponding mapping cost be $C_{k,p}$. Although we are dealing with only a subsequence T_p, the nature of the problem is essentially the same as finding the key time steps for the full sequence, where T_n, $R_{k,n}$ and $C_{k,n}$ are involved.

Suppose we have a constraint in T_p that the last key time step has to be time step p, i.e. $r_k = p$. Selecting k key time steps is then the same as selecting $k-1$ in the first m time steps, $m \in [k-1, p-1]$, and selecting one more from the rest. The overall mapping cost is the sum of the costs of two parts,

$$C_{k,p} = C_{k-1,m} + \parallel S_{m,p} \parallel \tag{7}$$

Let $R^*_{k,p}$ be the optimal selection for $R_{k,p}$, whose corresponding overall cost is $C^*_{k,p}$, the smallest among all possible $C_{k,p}$. We have the following optimal substructure: [12]

$$C^*_{k,p} = \begin{cases} \min\limits_{m \in [k-1, p-1]} \left(C^*_{k-1,m} + \parallel S_{m,p} \parallel \right) & \text{if } k > 2 \\ \min\limits_{m \in [1, p-1]} \left(\parallel S_{1,m} \parallel + \parallel S_{m,p} \parallel \right) & \text{if } k = 2 \end{cases} \tag{8}$$

To prove the correctness of Equation 8, it should be noted that if a key sequence is optimal, then its sub-sequence is also an optimal key time step selection for its corresponding subset of the full sequence. In other words, we could say if $R^*_{k,p} = \{r_1, r_2, ..., r_{k-1}, r_k\}$ is the optimal k key time steps selection from T_p, then its prefix $\{r_1, r_2, ..., r_{k-1}\}$ is also an optimal $k-1$ key time steps selection from $T_{r_{k-1}}$. This can be proved by contradiction: in the case of $k > 2$ in Equation 8, if $C^*_{k-1,m}$ is not the cost of the optimal mapping for the first m time steps, we could have replaced this sub-sequence by the optimal mapping with a lower $C_{k-1,m}$ value, and hence a lower $C_{k,p}$ from Equation 7, with a fixed $\parallel S_{m,p} \parallel$. This contradicts the facts that the mapping for $C^*_{k,p}$ is minimal.

In Equation 8, $C^*_{k,p}$ is the minimum of $C^*_{k-1,m}$ plus $\parallel S_{m,p} \parallel$ among the different selections of m. The $(k-1)$'th key time step in $R^*_{k,p}$ is

$$r^*_{k-1} = \begin{cases} \operatorname{argmin}\limits_{m \in [k-1, p-1]} \left(C^*_{k-1,m} + \parallel S_{m,p} \parallel \right) & \text{if } k > 2 \\ \operatorname{argmin}\limits_{m \in [1, p-1]} \left(\parallel S_{1,m} \parallel + \parallel S_{m,p} \parallel \right) & \text{if } k = 2 \end{cases} \tag{9}$$

In other words, r^*_{k-1} in Equation 9 is equal to the m that gives the minimum cost $C^*_{k,p}$ in Equation 8.

Based on Equations 8 and 9, the optimal key time steps $R^*_{k',p}$ can be solved by dynamic programming: We iteratively find k' key time steps, $k' = 2 \ldots k$, from the first p time steps $p = k' \ldots n$. Once the cost $C^*_{k',p}$ has been computed, this cost and the corresponding time step m will be stored in a 2D table indexed by k' and p. For $k' = 2$, the cost only relies on the segment mapping costs $\parallel S_{1,m} \parallel$ and $\parallel S_{m,p} \parallel$, $m = k'-1, \ldots, p-1$. For $k' > 2$, the cost is based on $C^*_{k'-1,m}$, $m = k'-1, \ldots, p-1$, which have been solved. Once the cost $C^*_{k,p}$ is obtained, the key time steps $R^*_{k,p}$ can be found by backtracking the time step stored in the table.

4.2 Segment Mapping Cost

To solve the above problem, we need to know the cost $\parallel S_{i,j} \parallel$. It can be defined differently for different use. In Liu's work, an energy function [12] is used to define the cost. Here to solve our problem of optimal mapping, we define $\parallel S_{i,j} \parallel$ as the cost of mapping from the sequence $S_{i,j} = \{i, ..., j\}$ to the key sequence $\{i, j\}$.

As proved in Section 4.1, if a key sequence is optimal, its subsequence is also an optimal selection of key time steps. Consequently, the mapping within each segment is also optimal. Given a dissimilarity metric $D_{i,j}$ for a pair of time steps i and j, which will be defined later. The mapping cost $\parallel S_{i,j} \parallel$ can be computed as the sum of the optimal mapping costs within the segment $S_{i,j}$.

$$\parallel S_{i,j} \parallel = \min_{i \le q < j} \left(\sum_{t=i}^{q} D_{i,t} + \sum_{t=q+1}^{j} D_{t,j} \right) \tag{10}$$

Time step q is a time step between time step i and time step j, as shown in Figure 3. Essentially the formula above computes the cost $\parallel S_{i,j} \parallel$ as the minimum cost for mapping the time steps before q to i, and the rest time steps to j among all the possible $q \in [i, j)$.

It should be noted that in certain simulation, the first a few time steps can have very small value during the initialization of the model. If the initial time step is undesired, the algorithm can be modified in order not to always choose the first time step. The basic idea behind this modification is to treat the first segment as a special case. Given the first segment from time steps 1 to j, its cost $\parallel S_{1,j} \parallel$ is computed by mapping all time steps to the j-th time step alone, other than both time steps 1 and j. i.e.

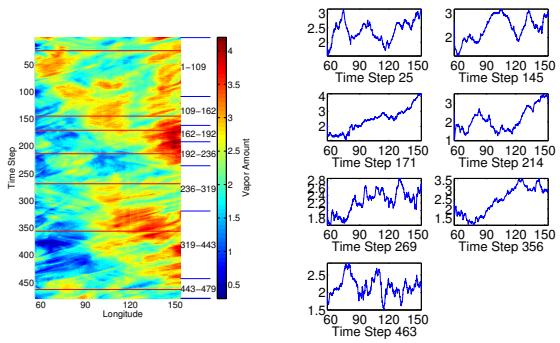
$$\parallel S_{1,j} \parallel = \sum_{t=1}^{j} D_{t,j} \tag{11}$$

Hereafter this special treatment is applied to all case studies in this paper.

For all the other cases when $S_{i,j}$ is not the first segment, $S_{i,j}$ is divided into two parts $S_{i,q}$ and $S_{q+1,j}$ by time step q, where the time steps in $S_{i,q}$ map to time step i, and the time steps in $S_{q+1,j}$ map to the last time step j, as shown in Figure 3. We define time step q as the *jump time step*, because at the point of time step q, each element in $S_{i,j}$ changes from mapping to time step i to mapping to time step j. No crossing is allowed in the mapping in our case considering the temporal coherence. $\parallel S_{i,j} \parallel$ is the minimum of the sum of the mapping cost in $S_{i,q}$ and $S_{q+1,j}$, for different choice of q. We may also view the solution to Equation 10 as finding the DTW warping between two sequences $S_{i,j}$ and $\{i, j\}$.

If the total number of time steps is n, then there are $\frac{n(n-1)}{2}$ pairs of (i, j), or possible segments. If we solve the cost for all the $\frac{n(n-1)}{2}$ segments, the results can be filled into a triangular matrix, which can be stored as input to the main algorithm described in Section 4.1.

Finally, $D_{i,j}$ is the dissimilarity metric between time step i and j. The metric can be defined differently for different goal. For example, if our focus is to compare two isosurfaces in two time steps, we can use the isosurface similarity map [3]. If we want to compare the distributions in two time steps, we can use Earth Mover's Distance [15] or K-L divergence [9]. Our algorithm does not depend on a particular kind of dissimilarity metric, so can be broadly applied to different problems.



(a) (b)

Figure 4: (a): Water vapor mixing ratio on Hovmoller diagram. (b): Water vapor mixing ratio on 7 key time steps marked as red lines in (a).

5 Results

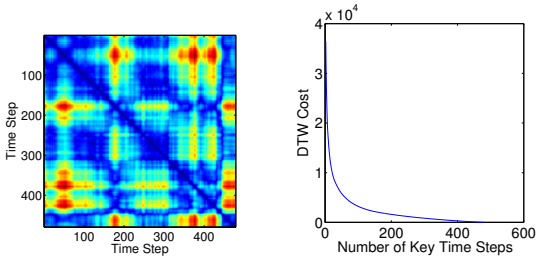
To test the efficacy of our algorithm, two case studies were conducted. The first case study is based on the data generated from a simulation of Madden-Julian Oscillation, a well known weather phenomenon, and the second case study is on the analysis of radiation from an astrophysics simulation.

5.1 Madden-Julian Oscillation

Madden-Julian Oscillation (MJO) is a weather phenomenon that consists of 30-60 days oscillation of surface and upper level winds in the tropical area near Indian and Pacific oceans. MJO can be characterized by an eastward progression of both enhanced and suppressed tropical rainfall. Study of MJO is important because it explains intra-seasonal variability in the tropics. MJO also influences the precipitation during the summer months in North America. The data set of the simulation consists of 479 time steps, where each time step contains of $2699 \times 599 \times 27$ voxels. The simulation was done by scientists in the Pacific Northwest National Laboratory [7].

Scientists usually observe MJO by Hovmoller diagrams generated from different variables, such as cloud intensity and water vapor. Figure 4(a) shows a Hovmoller diagram of the water vapor mixing ratio at the 583hPa pressure level . In the diagram, the X axis represents longitude, and the Y axis represents time. Each pixel in the diagram represents the average water vapor from points of different latitudes but a constant longitude and time step. Hovmoller diagram is commonly used for plotting meteorological data to highlight time progression of certain phenomena over a given spatial region [8]. The limitation of Hovmoller diagram is that it can only show a single spatial coordinate, e.g. longitude or latitude, at a time. If one desires to see the data in multiple dimensions, 2D or 3D space for example, Hovmoller diagram will be insufficient. In this case study, we use the sequence of scanlines from the Hovemoller diagram shown in Figure 4(a) as the time-varying data input. We detect salient time steps using our algorithm so that scientists can perform further analysis of the data based on the key time steps.

MJO can be observed from cloud movement along the longitude direction, and the cloud intensity is associated with water vapor mixing ratio. For this reason, we use the time-varying water vapor intensity data over a range of longitudes as the input to our algorithm to identify key time steps. We use the Earth Mover's Distance(EMD) to measure the dissimilarity of water vapor distribution between different time steps [15]. The EMD measures the minimum amount of work to fill a mass of earth into a collection of holes in the same space. The work is the sum of each unit of earth times the ground distance that it moves. EMD will assign a larger



(a) (b)

Figure 5: (a): Water vapor EMD dissimilarity map. (b) DTW cost with increasing number of key time steps for MJO data set.

dissimilarity value for two time steps if their peaks of water vapor are far away.

Figure 5(a) shows a dissimilarity matrix among the time steps using EMD for the water vapor data, where blue represents low and red represents high values. The large blue squares in the dissimilarity matrix along the diagonal line indicates there exists temporal coherence for data in adjacent time steps. Figure 5(b) shows the relationship between the DTW cost and the number of key time steps selected by our algorithm. It can be seen that the DTW cost drops sharply as more key time steps are selected. Applying our key time steps selection algorithm, we selected 7 key time steps. The plots of water vapor mixing ratio versus longitude for the key time steps are shown in Figure 4(b).

From these plots, we can see the movement of MJO over time. In the plots, MJO can be seen as peaks of the water mixing ratio, which is initially on the west (left in the plot) side of the domain. The peaks then move to the middle in key time step 2, and go to east with a large water vapor value in key time step 3. Before this MJO has not fully left the domain, a second MJO is formed in the east again, as shown in key time step 4. This is expected because MJO is a cyclic weather event. It can be seen that this peak moves to the middle in key time step 5, and to east in key time step 6. The small peak on the left of key time step 7 may indicate the formation of a third MJO.

In Figure 4(a), the selected key time steps are marked over the Hovmoller diagram by red lines, and the jump time steps are marked on the right side of Hovmoller diagram in blue. The distributions of the time steps between two adjacent jump time steps are very similar, and are represented by the key time step between them. There are more key time steps in the area of larger variance around time step 170, and fewer key time steps in the area of lower variance around time step 80 and time step 370.

5.2 Radiation of Astrophysics Turbulence

The astrophysics turbulence data set is from a three-dimensional radiation hydrodynamical simulation of ionization front instabilities [18]. The data set consists of 200 time steps, where each time step is a $600 \times 248 \times 248$ point regular mesh. The first star emits energetic UV radiation to the universe. The radiation ionized the surrounding gas to a temperature around $20,000K$. The distribution of the hot gas is interesting to the scientists, so we visualize the data with the isosurface of the temperature at $20,000K$, and see how it evolves change over time.

Among the 200 time steps in the dataset from which isosurfaces are computed, many of them look quite similar. This information redundancy makes it difficult for the user to focus on only the most essential information, in this case, how does the isosurface evolve. To assist the user, we apply our algorithm where the focus is to compare the isosurfaces and choose the most representative ones.

An important input to our algorithm is the dissimilarity between isosurfaces of all pairs across the entire time sequence. For this, we

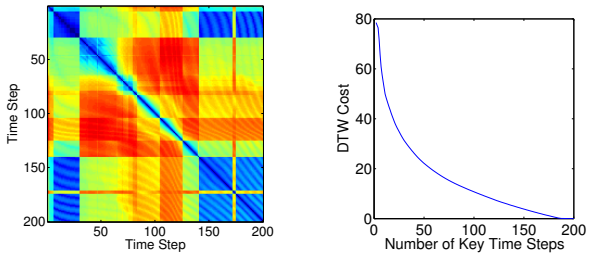


Figure 6: (a): Radiation isosurface dissimilarity map. (b) DTW cost drops with increasing number of key time steps for radiation data set.

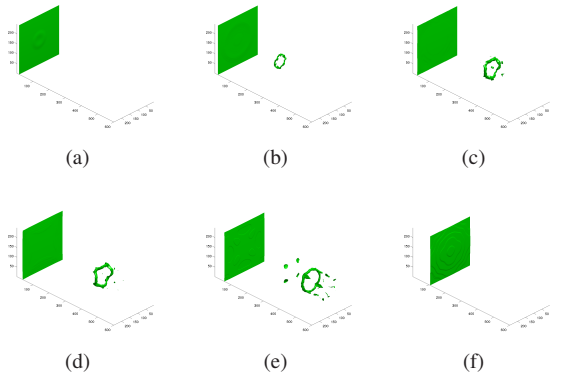


Figure 7: $20,000K$ isosurfaces of temperature field of 6 key time steps

use *isosurface similarity map* metric presented in [3]. In essence, this metric computes the mutual information between distributions of two distance fields derived from the isosurfaces. In our application, we use the similarities of all pairs of isosurfaces to build the 200×200 isosurface similarity map. After normalizing the similarity to a $[0,1]$ range, the value of one minus similarity is used to build the dissimilarity map input for our key time step selection algorithm.

The dissimilarity map is visualized in Figure 6(a). Blue color represents low dissimilarity, and red color represents high dissimilarity. The entries around the diagonal are mostly blue, which means the isosurface of one time step is similar to the isosurfaces in its adjacent time steps. Several different sized blue squares can be seen along the diagonal of dissimilarity map. Each square shows a group of similar isosurfaces.

Based on the dissimilarity map, we ran our key time steps selection program and selected 6 key time steps. Table 1 shows information about the selection results. In this table, each row is a segment $S_{i,j}$. It means that the time steps in this segment, time step i to time step j, map to the key time step, time step p^*, where $i \leq p^* \leq j$.

Table 1: 6 key time steps and the mapping for radiation isosurfaces

p^*	i	j	characteristics
24	1	30	one plane
48	31	65	a clear circle in front of the plane
78	66	83	small isosurfaces in front and back of the circle
93	84	104	small isosurfaces only in front of the circle
113	105	125	a broken cylinder in front of the plane
141	126	200	possible small isosurfaces in front of the plane

The segments listed in Table 1 show a good correspondence with the blue squares on the isosurface dissimilarity map in Figure 6(a). Thus, our algorithm is capable of making temporal classification of similar isosurfaces in a time-varying data set.

We visualized the 6 isosurfaces in Figure 7. Each of the 6 isosurfaces is representative for its corresponding time segment. We can see that they are quite different from each other, and each has its unique characteristics, as described in Table 1. The selected isosurfaces give the user a clear overview of how the isosurfaces evolve.

The relationship between the DTW cost and the number of key time steps selected is shown in Figure 6(b). The DTW cost does not drop as sharply as the one in our previous case study, the MJO data set, which is an indication that we may need more key time steps to represent the whole data set. However, limited by the screen space, we could not visualize too many key time steps at a time. In Section 6, we will describe an interactive key time steps browser, taking advantage of our algorithm, to explore time-varying data hierarchically.

5.3 Performance

A prototype of our algorithm was tested on a machine with Intel Core i7 2600 CPU, 16GB system memory, and nVidia GeForce GTX 560 GPU.

The dissimilarity matrix from the time-varying data is precomputed and used as the input to the dynamic programming algorithm. The performance and scalability of this preprocessing step depends on the dissimilarity metric we use. For the data set MJO, the dissimilarity matrix is computed based on the approximated EMD between the rows in the Hovemoller diagram, where each row represents data across different longitudes at a particular time step. Because data at each time steps are aggregated to a function of longitude, the computation of the dissimilarity matrix only took 25.6 seconds, including 24.2 seconds to generate the Hovemoller diagram from raw NetCDF data, and 1.4 seconds to compute EMD from the Hovemoller diagram using MATLAB. For the radiation data set, the dissimilarity computation first evaluates the distance from each grid point to each isosurface, which took 4.64 hours on GPUs for data at a full resolution. The mutual information computation is time-consuming, too, since joint histograms need to be constructed. For each pair of time steps, the mutual information is computed by scanning all voxels in their distance fields, which totally took 8.4 hours for 200 time steps. Thus, the total preprocessing time for radiation data set is $4.64 + 8.5 = 13.14$ hours.

The implementation of the dynamic programming algorithm has two major stages. The first stage is to run our dynamic programming algorithm to generate the time step index table. Given n time steps, the time complexity of our dynamic programming is $O\left(n^3\right)$. Figure 8 shows the time required to select different numbers of time steps from the MJO dataset. It can be seen that the algorithm took less than 0.25 second for all cases. Because the computation can be done very efficiently, the user can select different sub time interval and then re-compute the table interactively. Once the time step index table has been computed, the next stage is to select the key time steps of any user-desired number. Because the task of this stage essentially is to scan the table, the time spent on it is totally negligible compared to other stages. This allows the user to query different number of key time steps in real time.

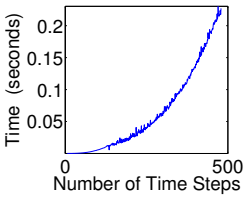


Figure 8: Performance of key time step selection for MJO.

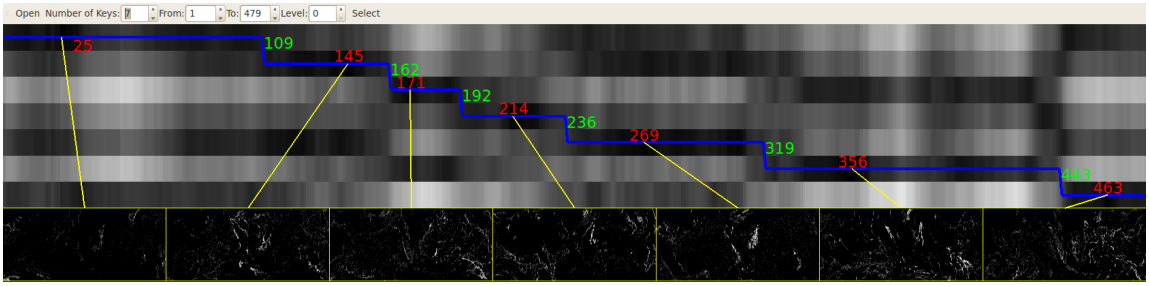


Figure 9: Time-varying data browser. Here 7 key time steps from all 479 time steps of the MJO data are selected. The top 7×470 2D map show the rows of the selected time steps from the original dissimilarity map. The data of the key time steps are rendered below the 2D map.

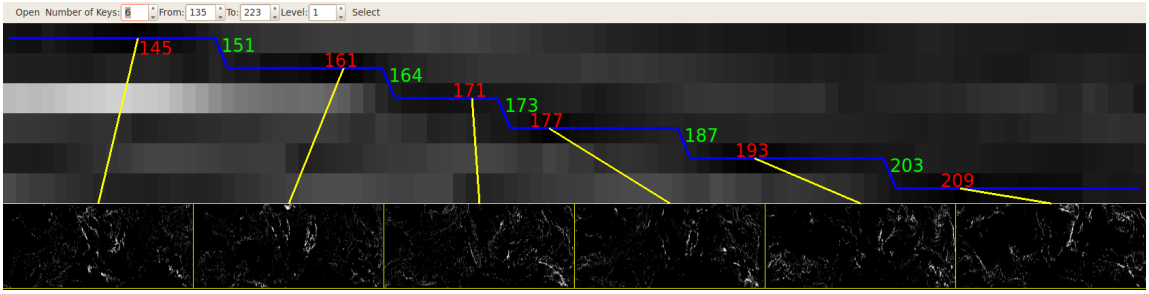


Figure 10: Zooming into a time interval in the browser. Here 6 key time steps are selected from the time step interval [135, 223] of the MJO data.

6 Key Time Steps based Time Varying Data Browser

To enable interactive analysis of time-varying data sets, we design a data browser that can take advantage of the information computed from our key time step selection algorithm. The key time steps and the mapping between the full sequence and the key sequence allow the user to navigate time-varying data at different levels of temporal detail. In the browser, we show information related to the nature of the time-varying data, including the dissimilarity matrix, the key time steps, the jump time steps, and the warping path between the full sequence and the key sequence. The data browser allows the user to interactively specify the desired number of key time steps, across the entire time sequence or within a local time segment. Figure 9 shows a snapshot of our system.

6.1 User Interface and Visual Display

A key component of this interface is the visualization of the dissimilarity matrix. It is visualized as a 2D greyscale image in the background. Each rectangle represents an entry of the matrix, where black represents low dissimilarity values, and white represents high dissimilarity values. Let n be the total number of time steps, and k be the number of key time steps. In the dissimilarity matrix viewer, we do not show the whole $n \times n$ dissimilarity matrix, but only the rows that correspond to the selected key time steps versus the full sequence in the columns, i.e. a $k \times n$ dissimilarity matrix. This matrix shows the dissimilarities between the key sequence and the full sequence.

From the results of key time steps selection, we establish a mapping between the key sequence and the full sequence. This mapping can be represented as a path in the dissimilarity matrix, shown as the blue lines in Figure 9. The warping path is composed of multiple horizontal lines segments. Each path segment is a mapping between one key time step to all its represented time steps.

In Figure 9, the key time step of each path segment is marked by a red time step index. The warping path between a pair of adjacent key time step is a time segment mentioned earlier in the algorithm section. Between a pair of adjacent key time steps, the warping path jumps from one row to the next row at the jump time step, which is marked by a green time step index. In order to provide detailed information for the time varying data set, visualization of data in the key time steps is shown below the dissimilarity map, each connected with the corresponding key time step shown on the warping path by a yellow line.

With this user interface, to browse the time varying data at different levels of detail, the user can interactively input the time interval of interest, and specify the number of desired key time steps. From the key time steps, the user may zoom in to a smaller time interval. They can repeat this process until the time steps that contain salient features are found.

6.2 Use Case for the MJO Data Set

Here we use the MJO data set described earlier in Section 5.1 to demonstrate the use of our browser. Given the pre-computed dissimilarity map, we pick 7 key time steps with our algorithm from the whole sequence, i.e., from time step 1 to time step 479.

Shown in Figure 9, the background is an image that shows a 7×479 dissimilarity matrix. 7 key time steps and 6 jump time steps are marked along the warping path. The 2D cloud intensity on the pressure level 850hPa for the 7 key time steps are displayed below the dissimilarity matrix, connected to their corresponding key time steps by yellow lines.

Since our algorithm run very efficiently after the dissimilarity map is computed, the selection of key time steps can be done interactively. By viewing the 2D cloud renderings at the key time steps, we found that time step 145 has high cloud intensity in the middle, which later moves to the southeast at time step 171. In time step 214, a second strong cloud appears in the southwest.

Assuming we are interested in the data around the three time steps mentioned above, we can drag an interval on the dissimilarity map, e.g. from time step 135 to time step 223. To get more details, we can double the number of key time steps in this interval to 6 key time steps. The new results are shown in Figure 10. Time step 145 and 171 are still the key time steps, with an additional key time step added in between. Time step 214, a key time step in the previous selection, is replaced by another 3 time steps, 177, 193, and 209.

The new key time step, time step 161, shows that before the middle point cloud moves to southeast, it first goes eastbound for some distance. Time steps 177, 193, and 209 show that this cloud wave stays at the east with a high intensity for a long time, and the second cloud wave is gradually formed in the southeast. We may further reduce the size of the time interval based on our interested key time steps to extract more detailed information of the MJO data set.

To summarize, with our system, the user can first obtain an overview of the entire data set with a small number of key time steps. From the key time steps, interesting time intervals can be selected where additional key time steps can be generated recursively until the desire features are found.

7 Limitation and Future Work

The main limitation of our work is that computing the dissimilarity matrix requires all time steps. Besides, because the complexity of distance computation for n time steps is $O(n^2)$, the distance computation can dominate the preprocessing stage. One example is our case study for radiation dataset, which took 4 hours on GPUs. In addition to apply different distance metrics and more efficient implementation, currently we are working on modifying the algorithm to make the dynamic programming algorithm work with a subset of the dissimilarity matrix. In such a case, we do not need to access the entire dataset, and thus our algorithm can be extended for in situ data reduction as well.

Another direction of our future work is to enhance our time-varying data browser. The current browser only displays a submatrix of the dissimilarity matrix. We believe that displaying the entire dissimilarity matrix to the user can provide additional useful information. For instance, by highlighting the time segments on the matrix, the user can visually evaluate whether the segments are sufficient to represent the whole data. Besides, the current design uses most of the screen space to display the dissimilarity matrix. Our new design will give more space to display the data in the selected time steps. This will allow the user to clearly compare the selected time steps side-by-side. One possible design is to shrink the dissimilarity matrix and arrange the time steps around the matrix. Also, the browser will have more user control. For instance, while currently the intensity of the dissimilarity map are automatically scales based on the current range of distance scope, the future design will allow the user to control the color mapping.

8 Conclusion

In this paper, we present a novel technique for selecting key time steps from large scale time-varying data sets. Our goal is to identify a subsequence from the entire time steps that can give a globally minimum cost. To achieve this goal, we apply Dynamic Time Warping to establish the mapping between the full and the subsequences, and choose the best subsequence among all the possible choices as the key time steps. The dynamic programming scheme we developed is very efficient, and thus can facilitate interactive data browsing. A time-varying data browsing system is designed to allow exploring the time varying data interactively in different levels of temporal scales. The algorithm was tested on a Madden-Julian Oscillation simulation data set using water vapor distributions over areas of different longitude. We also tested our algorithm on an astrophysics turbulence data set to select salient isourfaces over a long time sequence.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments. This work was supported in part by NSF grant IIS-1017635, US Department of Energy DOE-SC0005036, Battelle Contract No. 137365, and Department of Energy SciDAC grant DE-FC02- 06ER25779, program manager Lucy Nowell.

References

[1] H. Akiba, N. Fout, and K.-L. Ma. Simultaneous classification of time-varying volume data based on the time histogram. In *EuroVis '06: Proceedings of the Joint Eurographics - IEEE VGTC Symposium on Visualization 2006*, pages 171–178, 2006.

[2] H. Akiba and K.-L. Ma. A tri-space visualization interface for analyzing time-varying multivariate volume data. In *EuroVis '07: Proceedings of the Joint Eurographics - IEEE VGTC Symposium on Visualization 2007*, pages 115–122, 2007.

[3] S. Bruckner and T. Möller. Isosurface similarity maps. In *EuroVis'10: Proceedings of the Joint Eurographics - IEEE VGTC Symposium on Visualization 2010*, pages 773–782, 2010.

[4] 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.

[5] Y. Gu and C. Wang. Transgraph: Hierarchical exploration of transition relationships in time-varying volumetric data. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2015–2024, 2011.

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

[7] S. Hagos and L. R. Leung. Moist thermodynamics of the madden-julian oscillation in a cloud resolving simulation. *AMS Journal of the Atmospheric Sciences*, 24(21):5571–5583, 2011.

[8] E. Hovmöller. The trough-and-ridge diagram. *Tellus*, 1(2):62–66, 1949.

[9] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

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

[11] T.-Y. Lee and H.-W. Shen. Visualizing time-varying features with tac-based distance fields. In *PacificVis '09: Proceedings of IEEE Pacific Visualization Symposium 2009*, pages 1–8, 2009.

[12] T. Liu and J. R. Kender. Optimization algorithms for the selection of key frame sequences of variable length. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part IV 2002*, pages 403–417, 2002.

[13] A. Lu and H.-W. Shen. Interactive storyboard for overall time-varying data visualization. In *PacificVis '08: Proceedings of IEEE Pacific Visualization Symposium 2008*, pages 143–150, 2008.

[14] M. Müller. *Information Retrieval for Music and Motion*. Springer Verlag, 2007.

[15] Y. Rubner, L. Guibas, and C. Tomasi. The earth mover's distance, multi-dimensional scaling, and color-based image retrieval. In *Proceedings of the ARPA Image Understanding Workshop 1997*, pages 661–668, 1997.

[16] C. Wang, H. Yu, R. W. Grout, K.-L. Ma, and J. H. Chen. Analyzing information transfer in time-varying multivariate data. In *PacificVis '11: Proceedings of IEEE Pacific Visualization Symposium 2011*, pages 99–106, 2011.

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

[18] D. Whalen and M. L. Norman. "competition data set and description," in 2008 IEEE Visualization Design Contest. `http://vis.computer.org/VisWeek2008/vis/contests.html`, 2008.

[19] 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.

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

[21] J. Woodring and H.-W. Shen. Semi-automatic time-series transfer functions via temporal clustering and sequencing. *Computer Graphics Forum*, 28(3):791–798, 2009.