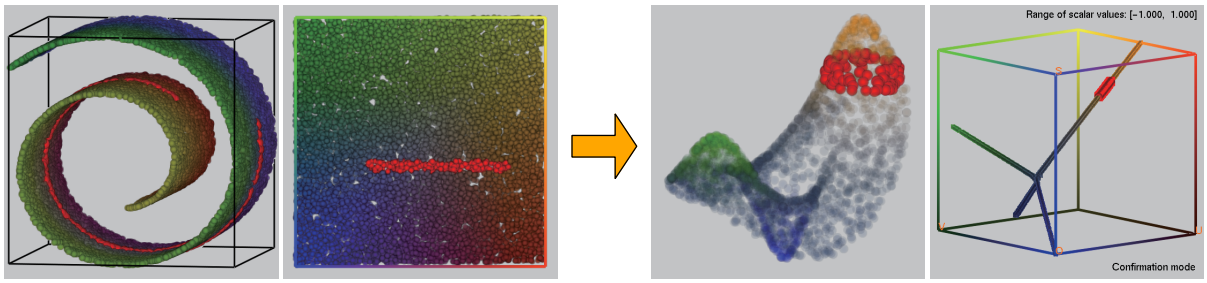
Applying Manifold Learning to Plotting Approximate Contour Trees

Shigeo Takahashi, *Member, IEEE*, Issei Fujishiro, *Member, IEEE*, and Masato Okada



(a) (b)

Fig. 1. A variant of an manifold learning technique allows us to extract contour trees: (a) The 3D distribution of the Swiss roll dataset (left) is elongated onto the 2D plane (right) through conventional nonlinear dimensionality reduction. (b) 3D scattered samples of a surface with two peaks and one pit (left) can be transformed into an approximate contour tree (right) using our approach. Different colors are assigned to the samples where the color of each axis represents the associated correspondence with their coordinates.

Abstract—A contour tree is a powerful tool for delineating the topological evolution of isosurfaces of a single-valued function, and thus has been frequently used as a means of extracting features from volumes and their time-varying behaviors. Several sophisticated algorithms have been proposed for constructing contour trees while they often complicate the software implementation especially for higher-dimensional cases such as time-varying volumes. This paper presents a simple yet effective approach to plotting in 3D space, approximate contour trees from a set of scattered samples embedded in the high-dimensional space. Our main idea is to take advantage of manifold learning so that we can elongate the distribution of high-dimensional data samples to embed it into a low-dimensional space while respecting its local proximity of sample points. The contribution of this paper lies in the introduction of new distance metrics to manifold learning, which allows us to reformulate existing algorithms as a variant of currently available dimensionality reduction scheme. Efficient reduction of data sizes together with segmentation capability is also developed to equip our approach with a coarse-to-fine analysis even for large-scale datasets. Examples are provided to demonstrate that our proposed scheme can successfully traverse the features of volumes and their temporal behaviors through the constructed contour trees.

Index Terms—Contour trees, manifold learning, time-varying volumes, high-dimensional data analysis.

1 Introduction

Extracting meaningful features from volumes and their temporal behaviors has been a key problem in scientific visualization, due to its ability to provide appropriate visualization parameters for a clear understanding of the input data. Among several representations of such features, *contour trees* have played an important role because they allow us to transform an input data into a skeleton-like tree, which delineates the topological evolution of isosurfaces according to the scalar field value. Several sophisticated algorithms have been proposed for constructing such contour trees together with application to several specific research topics in physics and biochemistry.

Formally speaking, existing algorithms can handle datasets of any dimension, while the input dataset is assumed to be samples of a single-valued function $s = s(p) = s(x_1, \ldots, x_n)$, where s represents a scalar field value and $p = (x_1, \ldots, x_n)$ indicates the coordinates of a n-dimensional data point. This function represents typical datasets such as terrain samples ($n = 2$), volume samples ($n = 3$), and its time-varying version ($n = 4$). These algorithms, however, generally complicate the associated software implementation especially when handling high-dimensional cases ($n \geq 4$). In addition, extracting contour trees from far higher-dimensional datasets has not been sufficiently tackled.

This paper presents a simple yet effective approach to extracting contour trees even from higher-dimensional datasets, by taking advantage of manifold learning. The manifold learning is a well-known dimensionality reduction scheme based on nonlinear transformations, which allows us to elongate the curvy distribution of the sample points and project it into a low-dimensional space, as shown in Fig. 1a. In our framework, the contour tree can be extracted just as a projection of point clouds into 3D space, without any complicated implementation except for introducing an available eigensolver. Thus, this is an approximate representation of the contour tree, while it is still expected to serve as an effective interface for exploring hyper-isosurfaces in the given high-dimensional dataset. For brevity, Fig. 1b shows a 2D result obtained through the use of our prototype system, where the corresponding portion of a 2D scalar (height) field (with two peaks and one pit) is highlighted by manually picking a point on the computed contour tree.

The overall process of the proposed approach consists of three primary computational steps (Fig. 2). First, we introduce a kernel-like neighborhood to each sample to seek the manifold proximity over the whole dataset. We then derive a dissimilarity matrix from the distance between every pair of sample points along the corresponding shortest path, in order to retrieve the underlying global structure of the manifold. Finally, we embed the manifold into the 3D space while respecting the calculated mutual distances among the samples. This is achieved by introducing a simple modification to an existing manifold learning technique while the associated results demonstrate that our approach is effective enough to explore the inner structures in multi-dimensional volumes. See Section 4 for details.

Since our framework relies on a numerical eigensolver, the size of the associated dissimilarity matrix should remain moderate. To make our framework scalable in terms of cardinality and dimensionality, we introduce hierarchical representations of the given samples. Starting from the original samples at the bottom level, we construct an interme-

- *Shigeo Takahashi and Masato Okada are with the University of Tokyo, E-mail: takahashis@acm.org, okada@k.u-tokyo.ac.jp.*
- *Issei Fujishiro is with Keio University, E-mail: fuji@ics.keio.ac.jp.*

diate level by adaptively merging two samples (or clusters) into a new cluster according to the closeness with respect to our new proximity metric. This amounts to adaptively redistributing a specific number of samples in the data domain using a *minimum spanning tree* strategy, while retaining the associated local proximity of the sample points. We then select a small number of landmark samples as the top level so that they are uniformly distributed according to the geodesic distance metric, which is also newly introduced to our framework for capturing the global structure of the underlying contour tree representation. In this respect, the present work allows us to extract the meaningful features at a visually-plausible resolution level even from the large-scale dataset. Moreover, interactive segmentation of the input data through the approximate contour tree can provide users with closeup shots of its minor topological structures. This capability is beneficial especially for analyzing high-dimensional cases because our approach eventually provides an interface for directly manipulating such invisible structures of the given datasets through the constructed approximate contour tree in an abstract low-dimensional space.

The remainder of this paper is organized as follows: Section 2 provides a brief survey on related research topics. Section 3 shows the overview of our approach, which is followed by the description of a basic algorithm for plotting approximate contour trees using a manifold learning technique in Section 4, and hierarchical representations for handling large-scale datasets with a currently available eigensolver in Section 5. Section 6 shows enhancements of the present approach for increasing its usability in a visualization environment. Section 7 presents several results generated by our prototype system, together with a discussion of features and limitations of the approach. Section 8 concludes this paper and refers to possible future extensions.

2 Related Work

Our work has been inspired by recent advancements in contour tree representation and manifold learning techniques.

2.1 Contour Trees

A contour tree is a special case of a *Reeb graph* [22], which represents a topological skeleton of a scalar function. Shinagawa et al. [25] originally introduced the concept of Reeb graph into the CG community for shape description purposes. Formally, a Reeb graph of a scalar function $s = s(x)$ is defined as a quotient space obtained by identifying different sample points if they belong to the same connected component of level sets $s^{-1}(C)$ where C is some scalar value. A contour tree is defined to be a Reeb graph that is free of cycles, and effectively captures the transitions of level sets when $s(x)$ is single-valued.

Contour trees [3] were initially invented for the analysis of terrain shapes in GIS applications to extract the topological transitions of the cross-sectional contours according to the height, and several algorithms for this computation have been presented [28] while their use was still limited. Bajaj et al. [1] introduced the contour trees to the scientific visualization community, as a truly attractive tool to visually explore the topological transitions of isosurfaces. The contour trees were further applied to several related problems including isosurface tracking [4], transfer function design [37, 30, 36], LoD control [29, 7], and spatial embedding analysis [31, 38].

For an algorithmic viewpoint, van Kreveld et al. [33] developed an optimal algorithm for 2D cases and a practical algorithm for 3D cases. Carr et al. [6] presented in his excellent algorithm, the state of the art in the contour tree construction for datasets of any dimension. Such algorithms had been limited to the extraction of the change in the number of connected components of level sets, until Pascucci et al. [19] presented an effective algorithm for detecting the change in their topological types such as the isosurface genus. High-order interpolants [5] have been tackled also for this purpose.

On the other hand, extracting a Reeb graph poses more complicated problems since it may contain cycles in itself. Shinagawa et al. [26] presented an initial algorithm for constructing a Reeb graph by identifying the correspondence between cross-sectional contours along the height axis. Cole-McLaughlin et al. [8] first presented an optimal algorithm for this purpose while the input data is limited to 2D manifold shapes. Recently, Pascucci et al. [21] presented the most sophisticated on-line algorithm for extracting the Reeb graph from a set of non-manifold simplicial decompositions.

Although existing algorithms are theoretically proven to extract contour trees (or Reeb graphs) from sample points of any dimension, the associated data structures are likely to become complicated especially for handling high-dimensional cases including time-varying volumes, due to severe limitations caused by the curse of dimensionality. Actually, existing methods [12, 17, 27, 16, 13] tackle the time-varying volumes by extracting a sequence of contour trees from their temporal snapshots and then tracking the transitions of the contour trees, since they rely on the premise that the sequence of isosurfaces at time steps is more informative than the hyper-isosurface of the original time-varying data. Note that Edelsbrunner et al. [11] presented a theoretical analysis of time-varying data by taking account of the coherence in Betti numbers.

However, computing contour trees directly from multi-dimensional volumes can still allow us to effectively identify their global structures at first glance. Our approach solves this problem by first analyzing a manifold structure induced by the distribution of sample points, and then elongating its curvy embedding in the data domain to find its projection into a low-dimensional space. In this aspect, our approach is similar to Weber et al.'s work [35] on finding a metaphor of volume topology as a topographic configuration. However, our approach differs in that it has high potential to overcome the curse of dimensionality through an effective coarse-to-fine analysis, while its associated implementation remains to be fairly simple even when handling high-dimensional datasets. Our approach can also be applied to the extraction of Reeb graphs, while our focus in this paper is basically on the contour tree extraction for the analysis of volumes and their temporal behaviors. Simple examples of higher-dimensional data analysis and Reeb graph construction will be exhibited in Section 7.

2.2 Manifold Learning

Manifold learning is a nonlinear dimensionality reduction scheme and has been intensively studied in recent years. The technique extracts manifold proximity inherited from the arrangement of the sample points embedded in a high-dimensional space, by constructing a proximity matrix among the samples in order to flatten the possible twisted embedding of the manifold in a low-dimensional space. Basically, manifold learning techniques are classified into two groups: local methods and global methods. The local methods first construct a similarity matrix by evaluating the closeness between each pair of samples, and then find their embeddings into a low-dimensional space by calculating the few smaller eigenvalues and their corresponding eigenvectors of the Laplacian matrix derived from the similarity matrix. This class of methods includes *Laplacian eigenmaps* [2] and *local linear embedding* [23]. On the other hand, the global methods employ a dissimilarity matrix instead, by estimating the distance between a pair of samples. *Isomap* [32] employs the geodesic distance along the shortest path between the samples in this context and retrieves the induced manifold proximity using *multidimensional scaling* (MDS) [9].

For calculating an approximate contour tree from the given samples, our approach employs the dissimilarity-based global scheme. This is because the contour tree is a quotient space obtained by identifying points in the same connected component on each level-set, and thus we have to plot different sample points at the same position on the contour tree in the projected space. However, the similarity-based scheme cannot represent such identity appropriately since we have to set some entries of the similarity matrix to be infinity in that case, which incurs severe problems in actual numerical computations. Indeed, our approach is a sophisticated variant of Isomap where the metrics for evaluating local proximity and global geodesic distances are replaced, in order to effectively represent a quotient space such as a contour tree.

3 Overview

Suppose that we try to extract an approximate contour tree from a given set of sample points (Fig. 2a). Our process for plotting the contour trees consists of six steps, which can be summarized as follows:

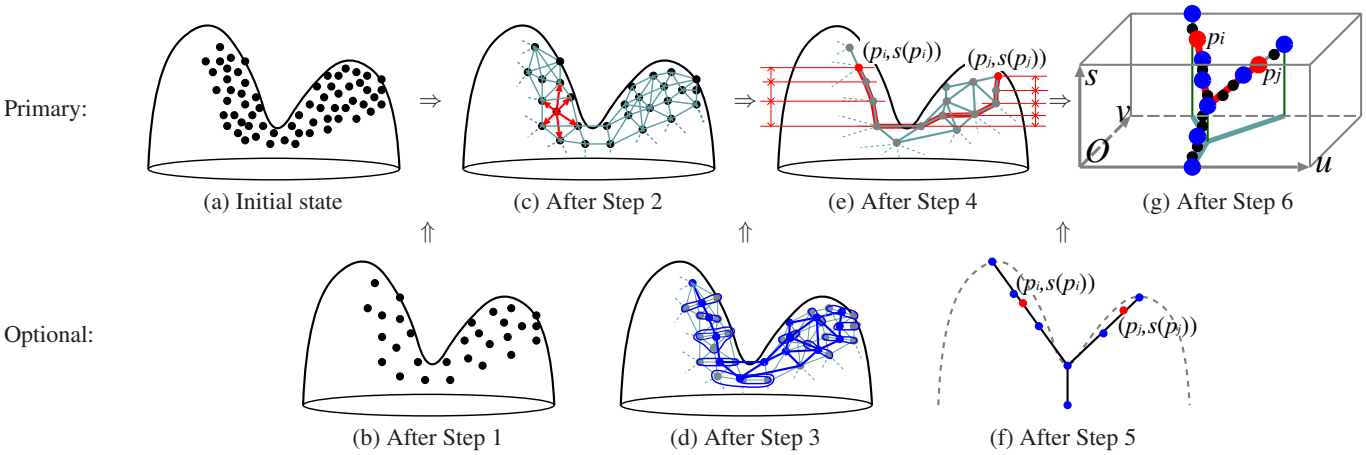


Fig. 2. Process for plotting approximate contour trees. (a) Initial samples. (b) A downsampled version (optional). (c) Manifold proximity constructed over the samples. (d) Hierarchical clusters of sample points (optional). (e) Accumulated difference in scalar field value along the shortest path. (f) Selection of landmark samples (optional). (g) An approximate contour tree embedded in 3D space. The upper row represents the primary flow of our approach while the bottom row depicts the optional steps for introducing hierarchical representation of data samples.

1. (Optional) Downsample the given set of sample points with respect to the data domain (Fig. 2b);
2. (Primary) Construct local manifold proximity of input sample points (Fig. 2c);
3. (Optional) Adaptively merge sample points to find a specific number of representative clusters of sample points (Fig. 2d);
4. (Primary) Calculate the distance (i.e. dissimilarity) between every pair of the sample points along the corresponding interpoint shortest path (Fig. 2e);
5. (Optional) Select a small number of landmark samples that are spaced uniformly over the underlying contour tree (Fig. 2f);
6. (Primary) Project the landmark samples into 3D space first and then locate the remaining samples by referring to the arrangements of projected landmark samples (Fig. 2g).

Note that Steps 2, 4, and 6 in the above process correspond to the primary three steps in the original Isomap framework, while Steps 1, 3, and 5 are optional ones that we newly introduced to construct the hierarchical representation of data samples for efficient computation. In what follows, without loss of generality, we explain our algorithm with an example of a height field in Fig. 2, and describe the primary steps in Section 4 and optional steps in Section 5, respectively.

4 Plotting Approximate Contour Trees

For plotting a given set of data samples to delineate the underlying contour tree in the Isomap framework, we have to evaluate the distance between each pair of samples by tracing the intermediate shortest path over the contour tree to be constructed. In our approach, we accomplish this by dividing the shortest path at intervening samples and accumulate the difference in scalar field value along the path, as shown in Fig. 2e. This is reasonable because we can locate any sample on the contour tree by referring to its relative position with respect to the other samples. This means that we can implement the above idea just by modifying the metric for evaluating geodesic distance between every pair of samples in the Isomap algorithm. Indeed, this is the core idea of this paper that faithfully follows the definition of the contour trees (Section 2.1). Furthermore, we also introduce a new metric for constructing the overall manifold proximity over the samples that allows us to contract the distribution of data samples effectively to a skeleton-like structure in the context of contour tree construction.

4.1 Constructing Proximity of Sample Points

Our first task is to retrieve the local proximity of the underlying low-dimensional manifold induced by the distribution of given sample points. Here, a manifold is thought of as a topological space which is locally flat. However, the overall shape of the manifold induced by the given samples is often curved intricately in the higher-dimensional data domain, and thus must be globally flattened out by tracking its local proximity over the sample points.

Suppose we have a set of m points $\{p_1, \ldots, p_m\}$, where $p_i \in \mathsf{R}^n (i = 1, \ldots, m)$ and its corresponding value of the scalar function $s(p_i) \in \mathsf{R} (i = 1, \ldots, m)$. This means that we take $(n+1)$-dimensional vectors $(p_i, s(p_i)) \in \mathsf{R}^{n+1} (i = 1, \ldots, m)$ as the input samples. For finding the local proximity, we introduce a kernel-like neighborhood to each sample point (Fig. 2c), and then construct a graph structure where its vertices represent the samples and an edge corresponds to the extracted proximity between some pair of the samples. The extracted proximity plays an important role in the next steps where we calculate the distance between sample points over the induced manifold. Note that we have no need to limit ourselves to the proximity derived from simplicial decomposition in this case; we can rather employ arbitrary proximity as shown in Fig. 2c.

For seeking adjacent points for each sample, we collect its first k-closest samples as adjacent points. However, for evaluating the closeness between a pair of samples, we do not use the ordinary Euclidean metric but employ the following new distance metric:

$$a|p_i - p_j|^{\alpha} + b|s(p_i) - s(p_j)|^{\beta}. \tag{1}$$

Here, we have decided to set $\alpha = 1$ and $\beta = 2$ while $a = b (= 1)$ empirically, after having tried several combinations of parameters. This is reasonable in the sense that we can easily discriminate between sample points if they have different scalar field values since the above metric is more sensitive to the difference in the scalar field value. In our computational scheme, this new metric effectively contracts sample points having similar scalar values to a single point on the approximate contour tree. Searching for the first k-closest samples is also appropriate because we can still construct the local proximity robustly over the sample points even when their distribution is not uniform within the data domain. Note that we can choose the number k arbitrarily, while we explore a relatively small number for k that expands the proximity graph over all the given samples with the help of our prototype system. Especially for grid samples, we can use $k = 8, 26$, and 80 for samples on 2D, 3D, and 4D data domains as its upper limit, respectively, as described by the work on contour tree extraction from digital images [18]. Basically, k should be larger as the corresponding

distribution of samples become more irregular. The choice of these parameters for constructing local proximity will be further discussed in Section 7.

4.2 Calculating Differences Between Sample Points

Having obtained a proximity graph over the sample points, we evaluate the difference between every pair of samples as their distance. For successfully elongating the manifold induced by the local proximity, we should calculate the *geodesic distance* along the corresponding shortest path passing over the induced manifold, rather than the simple Euclidean distance between the samples in the data domain. Recall that we employ Isomap [32] for this purpose, so that we can calculate the geodesic distance by accumulating the local interpoint distances along the corresponding shortest path. As described earlier, our distance metric for this purpose is just an accumulated difference in scalar field value along the shortest path. For example, suppose that the sequence of $(k+1)$ sample points $p_i, p_{\sigma(1)}, \ldots, p_{\sigma(k-1)}, p_j$ constitutes the shortest path between p_i and p_j, where σ represents some mapping to the IDs of sample points. The resultant geodesic distance between p_i and p_j can be obtained as the summation of the absolute interpoint differences in scalar field value:

$$|s(p_i) - s(p_{\sigma(1)})| + \cdots + |s(p_{\sigma(k-1)}) - s(p_j)|. \tag{2}$$

Note that the shortest path between every pair of samples can be easily identified by applying Dijkstra's shortest path finding algorithm to the proximity graph obtained in Step 2.

4.3 Embedding Samples into 3D Space

The above definition of the distance metric enables us to evaluate the relative positions of any pair of samples on a contour tree to be constructed. This means that we can embed the contour tree into a 2D plane while respecting such relative positions of every pair of samples, because the contour tree is a tree that can be embedded in a 2D plane without any self-intersections in nature. In the Isomap framework, we achieved this transformation using multidimensional scaling (MDS) [9] algorithm that fits the distribution of high-dimensional data samples into a low-dimensional space by optimizing the associated distortion in a least-square sense.

The MDS algorithm begins with constructing an $m \times m$ matrix D called a *squared-distance* (i.e., *dissimilarity*) *matrix*, where its (i, j)-entry D_{ij} represents the square of the geodesic distance from p_i to $p_j (1 \leq i, j \leq m)$. The projection of the samples can be preformed by transforming D to its normalized version called an *inner product matrix* B through the *double-centering formula*:

$$B = -\frac{1}{2} C D C^{\top},\ C = I_m - \frac{1}{m} 1_m 1_m{}^{\top}, \tag{3}$$

where I_m is an $m \times m$ identity matrix and 1_m is an $m \times 1$ column vector of m 1's. The projected coordinates of each sample are obtained by solving the eigenvalue problem of the normalized matrix B while we only need the first and second largest eigenvalues λ_1 and λ_2 and their corresponding eigenvectors e_1 and e_2. Actually, we can transform the sample p_i to its 2D coordinates: $(u_i, v_i) = (\sqrt{\lambda_1} e_{1i}, \sqrt{\lambda_2} e_{2i})$, where e_{ji} represents the i-th entry of the j-th largest eigenvector e_j.

While this is sufficient to find the projected coordinates of the sample $p_i (i = 1, \ldots, m)$, we also associate the scalar field value $s(p_i)$ with the projected coordinates (u_i, v_i) as its third coordinate in our approach as shown in Fig. 2g. This makes it possible to intuitively identify the topological transitions of hyper-isosurfaces according to the scalar field value. In this way, our sophisticated version of Isomap can successfully contract the distribution of the sample points in order to delineate the skeleton-like structure of the contour tree in the 3D space.

5 Hierarchical Representations

Since we have to solve the eigenvalue problem of the squared-distance matrix in the last primary step as described in the previous section, we like to keep the number of samples for calculating the matrix to be moderate without sacrificing too much accuracy. This idea is also justified by the fact that we cannot visually identify too detailed information obtained from a large number of samples. For the efficient computation even from a large number of samples, we introduce three optional hierarchical representations over the given samples, to enable adaptive sampling of the underlying contour tree.

5.1 Downsampling the Data Domain

Data samples for a time-varying volume often exceed the available capacity of memory space on a standard PC. Since every detail of the large dataset cannot be visually revealed on an ordinary computer display, we first reduce, as Step 1, the original number of time-varying samples with respect to the data domain (i.e., R^n) if necessary (Fig. 2b). This can be easily realized using iterative pyramid-like downsampling of data domain by a factor of 2 along each axis for grid samples, and random downsampling for scattered samples. This works well together with interactive segmentation of data samples to be equipped with in our system (cf. Section 6.2), since we can automatically restore the original density of samples through the coarse-to-fine analysis. In our implementation, we progressively reduce the number of samples until it becomes smaller than the predefined threshold, where we judged the threshold $160,000$ is a good trade-off between the computation time and accuracy.

5.2 Hierarchical Clustering of Samples

After having obtained the underling manifold proximity of data samples as a graph structure, we can introduce, as Step 3, different reduction of data sizes that adaptively samples over the underlying contour tree as the second hierarchical representation. In our approach, we implemented this hierarchy by referring to the progressive mesh scheme [15], where the edges of the mesh are iteratively contracted according to their weight values.

In our implementation, the weight for an edge should be defined to be the distance between the pair of two samples over the contour tree. Thus we employ the same metric as that for evaluating the geodesic distance, i.e., the difference in scalar field value (cf. Eq. (2)). Due to the definition of contour trees in Section 2.1, the metric will help us effectively contract sample points to a single point if they have almost the same scalar field values and stay spatially close to each other.

For the actual clustering of sample points, we first sort the list of edges in the proximity graph according to their weights in ascending order and fetch the edge at the top of the list. We then merge the corresponding two endpoints to create a new sample point as their cluster, and calculate its position as the barycenter of the contracted sample points (Fig. 2d). Successive application of these merging operations will provide a hierarchical clustering of sample points based on a minimum spanning tree strategy. In our implementation, the reduced number of clusters can be arbitrarily given by users, while it has been set to be $32,768 (= 2^{15})$ by default, which appears to be again an optimal trade-off between speed and loss of detail in our experiments.

5.3 Selecting Landmark Samples

As described in Section 4.3, we have to solve an eigenvalue problem to obtain the projection of contour tree in Step 6 of our process. This suggests that we can further reduce the computation time by restricting the size of the matrix to be as small as possible even when we utilize an available numerical solver for this problem. In our framework, this has been implemented as Step 5 by introducing the third hierarchical representation to the samples. In practice, we introduce a new hierarchical representation to select a specific number of sample points as the landmarks of the overall distribution of the given samples. Nonetheless, for better approximation of the underlying contour tree, the landmark samples should be arranged uniformly by referring to the relative geodesic distances obtained in Step 4.

Suppose that we have 8 samples together with the table of geodesic distances as shown in Fig. 3a, for example. Let us select #1 as the first landmark sample, while this can be arbitrarily chosen in this process. Geodesic distances from #1 to the other samples can be obtained just by referring to the first row of the above table, as shown in the top

	#1	#2	#3	#4	#5	#6	#7	#8
#1	0	5	8	5	7	3	1	6
#2	5	0	7	10	2	2	6	5
#3	8	7	0	13	9	5	9	2
#4	5	10	13	0	12	8	4	11
#5	7	2	9	12	0	4	8	7
#6	3	2	5	8	4	0	4	3
#7	1	6	9	4	8	4	0	7
#8	6	5	2	11	7	3	7	0

(a)

	#1	#2	#3	#4	#5	#6	#7	#8
#1	(0)	5	8	5	7	3	1	6
d_1	-	5	8	5	7	3	1	6
#3	(8)	7	(0)	13	9	5	9	2
d_2	-	5	-	5	7	3	1	2
#5	(7)	2	(9)	12	(0)	4	8	7
d_3	-	2	-	5	-	3	1	2
#4	(5)	10	(13)	(0)	(12)	8	4	11
d_4	-	2	-	-	-	3	1	2

(b)

Fig. 3. Selecting landmark samples that are uniformly distributed over the contour tree. (a) The table of relative geodesic distances between 8 samples. (b) The updates of the distance of each sample from its closest landmark sample.

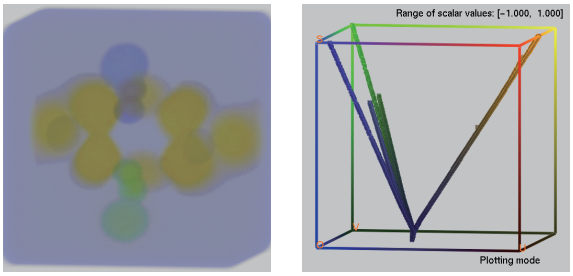


Fig. 4. Direct volume rendering of the "neghip" dataset (65^3 samples) with point splatting (left) and its corresponding approximate contour tree (right).

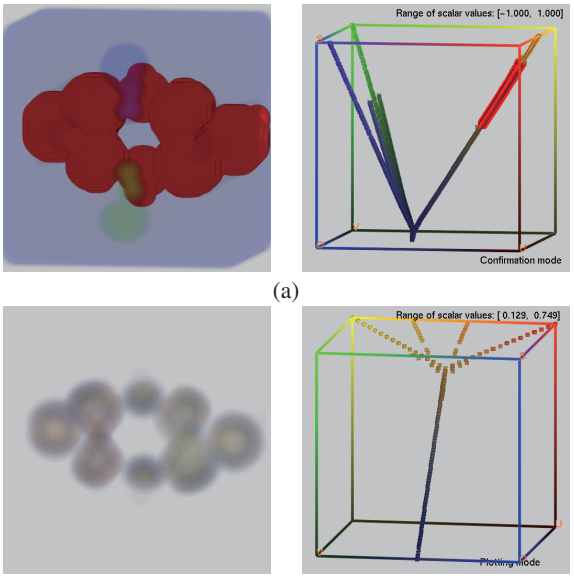


Fig. 5. Interactive segmentation with an approximate contour tree for LoD analysis: (a) Specifying a subvolume with the contour tree, and (b) a more detailed analysis of the segmented subvolume in red in (a).

row of Fig. 3b. Let d_i denote the list of geodesic distances of each sample from its closest landmark sample after the i-th landmark selection. This means that d_1 is equivalent to the list of geodesic distances from #1 (Fig. 3b). According to this result, we select #3 as the second landmark sample because it is the most distant sample from the previous landmark sample #1. We then seek the list of distances from #3 by referring to the third row of the table in Fig. 3a, and compose the list d_2 in such a way that each of its entry represents the distance from the closest landmark sample(s). This can be obtained by comparing the corresponding entries of d_1 and d_2, and choosing the smaller distance for each sample (Fig. 3b). In the same way, we can obtain #5 as the next landmark sample. Repeating this process allows us to collect a specific number of landmarks, while the number can be specified again by users and has been set $256(=2^8)$ empirically by default in our implementation. Note that we have no need to update the distances once the corresponding samples have been elected as the landmarks (Fig. 3b).

When we employ Step 5 to select the landmark samples, our process of projecting samples into 2D plane becomes two-fold: we first project the landmark samples into 2D plane, and then calculate the 2D positions of the remaining samples by referring to the 2D distribution of the projected landmarks. This is accomplished by employing the landmark MDS [10], which is computationally more efficient than the original MDS algorithm. Readers may refer to [10] for more details.

6 Usability Enhancement

This section focuses on several enhancements of the present approach especially for increasing its usability in a visualization environment.

6.1 Direct Volume Rendering

A given set of samples is appropriately projected into the 3D space (u,v,s) in the sense that two samples are geometrically close to each other in that space if they are topologically close also on the corresponding contour tree. This leads us to an idea that we assign optical attributes such as color and opacity to the samples according to their geometric positions in that space, so that we can visually recognize how different the given two samples are by referring to their resultant colors. This naturally results in the accentuation of the topological structure of the given dataset for better analysis of its content, and allows us to equip our framework with a simple yet effective way of designing transfer functions in the phase of direct volume rendering, when compared with the existing sophisticated design schemes [31, 36]. In our implementation, we employ the RGYB color geometry [34] to the horizontal uv-square region where the red-green and yellow-blue color axes are aligned along the two diagonals. We further assign the opacity channel to the vertical axis s to make sample points of larger scalar field values more visible in the rendered images.

Fig. 4 shows a visualization result of the "neghip" dataset, where the spatial probability distribution of the electrons in a high potential protein molecule is simulated. We employed point splatting for our rendering scheme where each point is rendered as a Gaussian texture, and assigned different colors to the samples according to the above setting of optical attributes. Note that, in the figure, the color legend of the cubical frame on the right indicates the correspondence between the position of a plot in the cube and the color to be assigned.

6.2 Interactive Segmentation

The adaptive sampling efficiently limits the size of the eigenvalue problem to be moderate while we cannot extract local details of the contour tree since MDS is likely to identify the global structure of the given data in nature. This implies that closeup analysis of the contour tree will be helpful in finding its local branching structures at an appropriate level of detail (LoD). For this purpose, our prototype system makes it possible to segment the given set of sample points by interactively specifying the region of the contour tree.

Fig. 5 shows how we can carry out such segmentation of volume data in our system. First we specify a partial region of the contour tree we are interested in as shown in Fig. 5a, and then conduct the segmentation. The sample points involved in the segmented region can be further analyzed to find the local details of the contour tree as shown in Fig. 5b. Note that this segmentation is still possible even with the aforementioned hierarchical representations since our system retains the mapping from a representative plot on the contour tree to the original sample points. This closeup operation compensates for the loss of the local details in our approximate representation of contour trees, and further provides an effective LoD capability especially for the analysis of large-scale datasets.

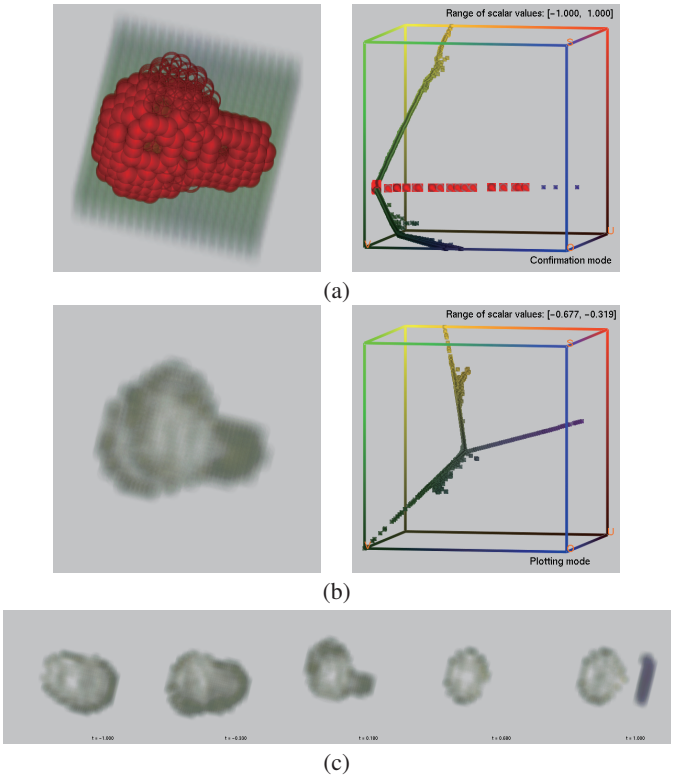


Fig. 6. Proton and hydrogen-atom collision ($80^3 \times 16$ samples): (a) Global analysis with a contour tree, (b) the segmented time-varying volume, and (c) its temporal behavior. The interaction between the proton and hydrogen-atom has been successfully identified.

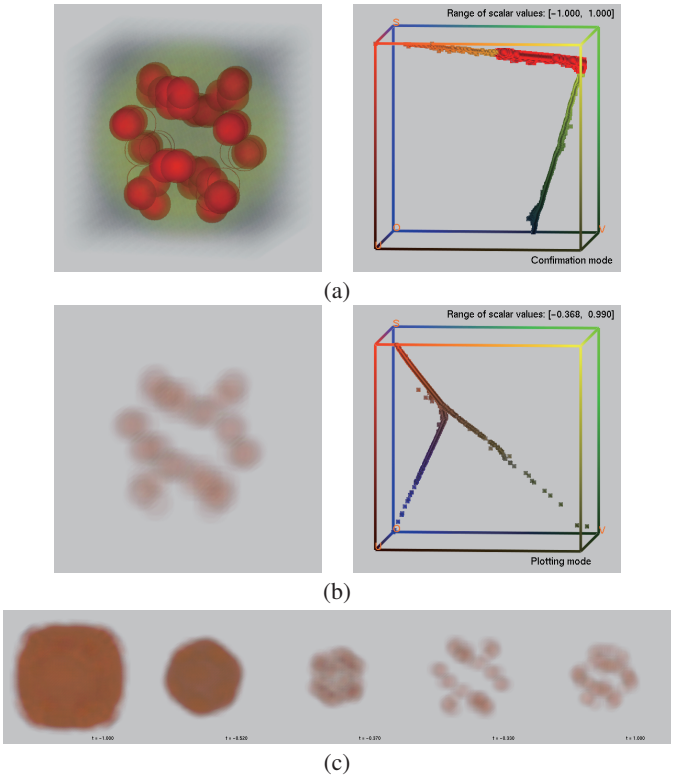


Fig. 7. Implosion in laser fusion ($64^3 \times 16$ samples): (a) Global analysis with a contour tree, (b) the segmented time-varying volume, and (c) its temporal behavior. Complicated temporal behavior of the contact surface between the inner fuel and outer pusher has been detected.

7 Results and Discussions

This section provides several experimental results in order to demonstrate the capability of our approach. Our prototype system has been implemented on a laptop PC with Intel Core2Duo T9500 CPUs running at 2.60GHz and 4GB RAM. All the experiments presented here were conducted on this computational environment. Note that in our implementation we employed Boost Graph Library for representing proximity graphs and a numerical library SLEPc [14] to solve the eigenvalue problems. To search for the nearest neighbors, we employed a kD-tree data structure in our implementation.

7.1 Results

We tested our approach with two time-varying volume datasets. Fig. 6 shows the result of a time-varying volume dataset where the proton and hydrogen-atom collision is simulated [30]. Since a time-varying dataset usually contains a large number of samples, our system subsampled the original resolution $80^3 \times 16$ to $20^3 \times 16$ in Step 1 (cf. Section 5.1) while the system automatically increases the resolution up to the original once feature subvolumes have been specified in the analysis. Fig. 6a shows a snapshot where the global analysis of the time-varying volume was conducted with an approximate contour tree. By segmenting the branch of the tree, we can further examine the details of the segmented volume that corresponds to the interaction between the proton and hydrogen-atom as shown in Fig. 6b. Tracking the temporal behavior of this subvolume provides us with significant features of this time-varying dataset as shown in Fig. 6c, where the positive charge of the proton significantly influences the behavior of an electron around the hydrogen-atom.

Fig. 7 exhibits the other example where the implosion in laser fusion is simulated [24]. Here, the corresponding time-varying data contains $64^3 \times 16$ voxels. Again, in this example, the system reduced its size to $16^3 \times 16$ in Step 1 while it automatically retrieves the original resolution again when the user goes into the analysis of local details. In this example, it is important to track the temporal behavior of the contact surface between the inner fuel ball and outer pusher during the stagnation phase. In the global analysis of the given volume (Fig. 7a), we can locate the contact surface at a top branch in the constructed approximate contour tree. By extracting a subvolume corresponding to the branch (Fig. 7b), we can identify the timing of implosion between the third and fourth snapshots through the temporal transition (Fig. 7c).

We also analyzed the scattered samples of the 7D blobby function $f : \mathrm{R}^7 \rightarrow \mathrm{R}$, which consists of a weighted sum of Gaussian functions and has two maxima and two minima in the target 7D data domain as shown in Fig. 8a[1]. Even from this high-dimensional samples, we can successfully extract the approximate contour tree as shown in Fig. 8b, which reveals the potential of the present approach.

Table 1 tabulates the computation times required for the analysis of the datasets used in our experiments.

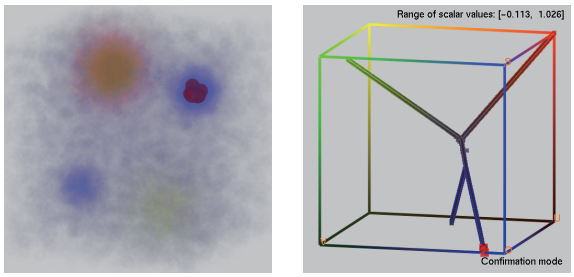
7.2 Discussions

Correctness of approximate contour trees: Our approach to contour tree construction is simple to implement as compared with any other existing approaches, and yet powerful enough to analyze high-dimensional cases including time-varying datasets. Indeed, in Fig. 9, the "neghip" (Fig. 9a) and "nucleon" (Fig. 9b) datasets were chosen to demonstrate that our manifold learning-based approach can extract the approximate contour trees, which are substantially identical to those extracted by Carr's graph-based approach [6] except for a small number of minor details. Note that the graph-based representations were obtained in a *fine-to-coarse* fashion in that all the local features were extracted first, and then the global skeleton was obtained through graph simplification process [7, 20, 29].

[1]The formula of this function appears as a supplemental material.

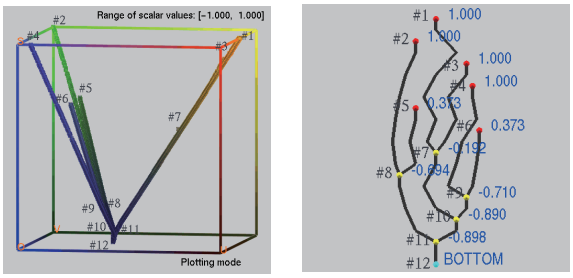
Table 1. Computation times. (in seconds)

Dataset	Dim.	# of samples	k	Step 1 Downsample	Step 2 Proximity	Step 3 Cluster	Step 4 Distance	Step 5 Landmark	Step 6 Project	Total
Nucleon	3D	$68,921 (= 41^3)$	14	NA	5.44	4.21	14.09	3.74	1.82	29.30
Neghip	3D	$274,625 (= 65^3)$	26	NA	42.54	82.17	23.27	2.79	1.21	151.98
Collision	4D	$8,192,000 (= 80^3 \times 16)$	64	8.57	48.68	639.38	24.24	5.54	1.20	727.63
Implosion	4D	$4,194,304 (= 64^3 \times 16)$	42	4.38	20.09	82.06	20.26	5.20	1.22	133.21
Blobby	7D	32,768	14	NA	10.36	NA	11.16	3.04	1.04	25.60

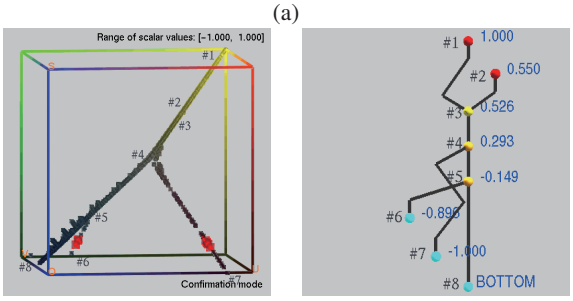


(a) (b)

Fig. 8. Analyzing scattered samples of a 7D function (32,768 samples): (a) Rendered image of the samples in the space spanned by the first three coordinates, and (b) its corresponding approximate contour tree.



(a)



(b)

Fig. 9. Comparison with a graph-based approach [6]: (a) "Neghip" (65^3 samples) and (b) "Nucleon" (41^3 samples). The left images show approximate contour trees extracted with our approach while the right show those obtained by the graph-based approach [6]. The labels indicate the correspondence between the critical points while numerical values (in blue) represent the normalized scalar field values.

Coarse-to-fine vs. Fine-to-Coarse: The contour trees obtained using our framework are only approximate representations while users can freely select specific parts of the trees to investigate local subvolumes at finer levels. This *coarse-to-fine* analysis can successfully cut out significant features even from high-dimensional data and bring out its visual appearance in 3D space. The present approach employs an eigensolver for locating the representative samples on the approximate contour tree, and thus the size of the samples should be kept moderate. Nonetheless, the eigenanalysis-based scheme can extract the macroscopic structure of the representative samples only from their mutual positional relationships over the underlying manifold, without regard to minor local details. This is also justified by the fact that our visual ability to recognize features at a glance has its own limit in terms of spatial frequency. Therefore, the information drill-down approach to zooming up specific microscopic features on demands would be more effective in our context (Fig. 5).

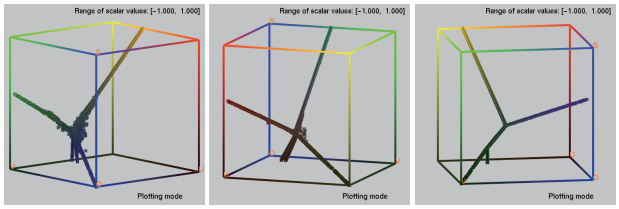
Distribution of input samples and neighborhood definition: The input data is assumed to be uniformly distributed over the underlying manifold in our approach, while irregular samples can also be handled since we introduced Eq. (1) as the new metric to evaluate the local proximity. If we use the ordinary Euclidean metric instead, we cannot fully contract the distribution of samples to a contour tree around its branching parts as shown in Fig. 10a. This is because, with the ordinary Euclidean metric, different connected components at the same scalar level are likely to be merged by way of such branching parts during the search for the k-closest neighbors. The coordinates of the samples with respect to each axis are normalized in the above results while we may have to use different normalization to spatial and temporal dimensions. The results are also influenced by the number k of the nearest neighbors. Basically, as shown in Figs. 10b and 10c, the approximate contour tree becomes fat as k decreases while some branches or cycles may also be excessively contracted if k becomes large. See also Fig. 1b ($k = 14$) for comparison. Our system optionally provides us with a means of semi-automatically finding an appropriate number for k, since we can reduce the computation time for the adaptive clustering when k is relatively small as shown in Table 1.

Ability to extract Reeb graphs: In addition, as described in Section 2, our method can extract Reeb graphs from surfaces with nonzero genus as well. Fig. 11 shows an example of the extracted Reeb graph obtained from a teapot-like object using our prototype system. Highlighting the corresponding portion in red helps us visually confirm that the complicated topological transition of cross-sectional contours according to the height was correctly captured in our system. This example shows that our framework is applicable to surfaces with nonzero genus while the cycle-induced unmatured layout and colorization of projected samples could still be improved.

8 Conclusion

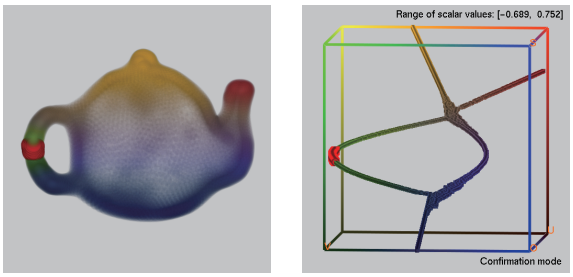
This paper has presented an approach to plotting approximate contour trees by embedding a set of sample points into a low-dimensional space using a variant of Isomap. Our process for the contour tree construction consists of extracting local manifold proximity among the sample points, flattening the curvy embedding of the manifold in the high-dimensional space, and projecting it into the low-dimensional space for our visual analysis. Our contribution lies in the sophistication of the ordinary distance metric of Isomap, which actually provides us with an effective means of contracting sample points to the contour tree. Step-by-step hierarchical representations enable us to shift our reference space from the data domain to contour tree when adaptively resampling the original distribution of samples, which keep the size of the problem to be moderate for reducing the computational cost.

Our future extension includes finding more attractive embeddings of contour trees into 3D space. Extracting the explicit graph structure from the approximate contour tree representation is beneficial for further quantitative analysis of extracted topological features. If we come up with other distance metrics to contract high-dimensional scientific datasets, it will provide more effective ways to understand the contents of the given datasets.



(a) (b) (c)

Fig. 10. Choice of parameters for proximity metric: Approximate contour trees obtained using (a) the Euclidean distance metric for local proximity, (b) an insufficient number of k-neighbors ($k = 6$), and (c) an excessive number of k-neighbors ($k = 80$). See Fig. 1b ($k = 14$) for comparison.



(a) (b)

Fig. 11. Analyzing surfaces with non-zero genus: (a) A teapot-like object (5,600 samples) and (b) its approximate Reeb graph.

Acknowledgments

We thank Reiji Suda for his guidance in exploring numerical eigensolvers. Haruhisa Ishida and Jun Kobayashi helped us implement an early version of the prototype system. We also thank anonymous reviewers for their valuable comments. This work has been partially supported by Japan Society of the Promotion of Science under Grants-in-Aid for Scientific Research (A) No. 20240020, Scientific Research (B) No. 18300026, Challenging Exploratory Researches No. 21650019 and No. 20650010, and Young Scientists (B) No. 17700092.

References

[1] C. L. Bajaj, V. Pascucci, and D. R. Schikore. The contour spectrum. In *Proc. IEEE Vis.'97*, pages 167–173, 1997.

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps for spectral embedding and clustering. *Neural Computations*, 15(6):1373–1396, 2003.

[3] R. L. Boyell and H. Ruston. Hybrid techniques for real-time radar simulation. In *IEEE Proc. Fall Joint Computer Conf.*, pages 445–458, 1963.

[4] H. Carr and J. Snoeyink. Path seeds and flexible isosurfaces using topology for exploratory visualization. In *Proc. Joint Eurographics-IEEE TCVG Symp. Visualization 2003*, pages 49–58, 285, 2003.

[5] H. Carr and J. Snoeyink. Representing interpolant topology for contour tree computation. In *Topology-Based Methods in Visualization II*, pages 59–73. Springer, 2009.

[6] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry: Theory and Applications*, 24(2):75–94, 2003.

[7] H. Carr, J. Snoeyink, and M. van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *Proc. IEEE Vis. 2004*, pages 497–504, 2004.

[8] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in Reeb graph of 2-manifolds. In *Proc. 19th ACM Symp. Computational Geometry*, pages 344–350, 2003.

[9] T. Cox and M. Cox. *Multidimensional Scaling*. Chapman & Hall, London, 1994.

[10] V. de Silva and J. B. Tanenbaum. Global versus local method in nonlinear dimensionality reduction. In *Proc. Neural Information Processing Systems Conf. 2002*, pages 705–712, 2002.

[11] H. Edelsbrunner, J. Harer, A. Mascarenhas, and V. Pascucci. Time-varying Reeb graphs for continuous space-time data. In *Proc. 20th ACM Symp. Computational Geometry*, pages 366–372, 2004.

[12] I. Fujishiro, T. Azuma, Y. Takeshima, and S. Takahashi. Volume data mining using 3D field topology analysis. *IEEE CG&A*, 20(5):46–51, 2000.

[13] I. Fujishiro, R. Otsuka, S. Takahashi, and Y. Takeshima. T-map: A topological approach to visual exploration of time-varying volume data. In *Proc. 6th Int'l Symp. High Performance Computing*, volume 4759 of *Springer LNCS*, pages 176–190, 2008.

[14] V. Hernandez, J. E. Roman, A. Tomas, and V. Vidal. SLEPc users manual. Technical report, Technical Univ. of Valencia, Spain, 2004.

[15] H. Hoppe. Progressive meshes. In *Proc. SIGGRAPH96*, pages 99–108, 1996.

[16] P. Keller and M. Bertram. Modeling and visualization of time-varying topology transitions guided by hyper Reeb graph structures. In *Proc. Computer Graphics and Imaging (CGIM) 2007*, 15–20.

[17] A. Mascarenhas and J. Snoeyink. Implementing time-varying contour trees. In *Proc. 21st ACM Symp. Computational Geometry*, pages 370–371, 2005.

[18] S. Mizuta and T. Matsuda. Description of digital images by region-based contour trees. In *Proc. Int'l Conf. Image Analysis and Recognition 2005*, volume 3656 of *Springer LNCS*, pages 549–558, 2005.

[19] V. Pascucci and K. Cole-McLaughlin. Efficient computation of the topology of level sets. In *Proc. IEEE Vis. 2002*, pages 187–194, 2002.

[20] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. Multi-resolution computation and presentation of contour trees. In *Proc. IASTED Conf. Visualization, Imaging, and Image Processing*, 2004. 452-290.

[21] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas. Robust on-line computation of Reeb graphs: Simplicity and speed. *ACM TOG*, 26(3):58, 2007.

[22] G. Reeb. Sur les points singuliers d'une forme de pfaff completement integrable ou d'une fonction numerique. *Comptes Rendus Acad. Sciences Paris*, 222:847–849, 1946.

[23] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[24] H. Sakagami, H. Murai, Y. Seo, and M. Yokokawa. 14.9 tflops three-dimensional fluid simulation for fusion science with HPF on the earth simulator. In *Proc. ACM/IEEE SC2002*, 2002.

[25] Y. Shinagawa, Y. L. Kergosien, and T. L. Kunii. Surface coding based on Morse theory. *IEEE CG&A*, 11(5):66–78, 1991.

[26] Y. Shinagawa and T. L. Kunii. Constructing a Reeb graph automatically from cross sections. *IEEE CG&A*, 11(6):44–51, 1991.

[27] B.-S. Sohn and C. Bajaj. Time-varying contour topology. *IEEE TVCG*, 12(1):14–25, 2006.

[28] S. Takahashi, T. Ikeda, Y. Shinagawa, T. L. Kunii, and M. Ueda. Algorithms for extracting correct critical points and constructing topological graphs from discrete geographical elevation data. *Computer Graphics Forum*, 14(3):181–192, 1995.

[29] S. Takahashi, G. M. Nielson, Y. Takeshima, and I. Fujishiro. Topological volume skeletonization using adaptive tetrahedralization. In *Proc. Geometric Modeling and Processing 2004*, pages 227–236, 2004.

[30] S. Takahashi, Y. Takeshima, and I. Fujishiro. Topological volume skeletonization and its application to transfer function design. *Graphical Models*, 66(1):22–49, 2004.

[31] Y. Takeshima, S. Takahashi, I. Fujishiro, and G. M. Nielson. Introducing topological attributes for objective-based visualization of simulated datasets. In *Proc. Volume Graphics 2005*, pages 137–145, 236, 2005.

[32] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[33] M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *Proc. 13th ACM Symp. Computational Geometry*, pages 212–220, 1997.

[34] C. Ware and W. Cowan. The RGYB color geometry. *ACM TOG*, 9(2):226–232, 1990.

[35] G. H. Weber, P.-T. Bremer, and V. Pascucci. Topological landscapes: A terrain metaphor for scientific data. *IEEE TVCG*, 13(6):1416–1423, 2007.

[36] G. H. Weber, S. E. Dillard, H. Carr, V. Pascucci, and B. Hamann. Topology-controlled volume rendering. *IEEE TVCG*, 13(2):330–341, 2007.

[37] G. H. Weber, G. Scheuermann, H. Hagen, and B. Hamann. Exploring scalar fields using critical isovalues. In *Proc. IEEE Vis. 2002*, pages 171–178, 2002.

[38] X. Zhang and C. Bajaj. Extraction, visualization and quantification of protein pockets. In *Proc. 6th Annual Int'l Conf. Computational System Bioinformatics (CSB2007)*, pages 275–286, 2007.