

# Identifying Nearly Equally Spaced Isosurfaces for Volumetric Data Sets

Martin Imre<sup>a</sup>, Jun Tao<sup>a</sup>, Chaoli Wang<sup>a</sup>

<sup>a</sup>Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556

---

## Abstract

Isosurfaces are an important visual representation of volumetric data sets and isosurface extraction and rendering remains one of the most popular methods for volume visualization. Previous works identify a small set of representative isosurfaces from a set of sample ones, providing a concise description of the underlying volume. However, these methods do not lend themselves to equally spaced isosurfaces, i.e., keeping the same distance between neighboring isosurfaces, which can be advantageous from the user's perspective in terms of visual summarization and interactive exploration. In this paper, we present a new solution that efficiently identifies a set of nearly equally spaced isosurfaces for a given volume data set. Our approach includes an estimation stage of linear interpolation and a refinement stage of binary search in order to balance the tradeoff between quality and performance. The refinement stage can incorporate spike and/or jump treatments to possibly improve the convergence. Experimenting with multiple data sets of different sizes and characteristics, we perform both quantitative and qualitative studies, demonstrate the efficiency and effectiveness of our approach, and summarize our findings.

**Keywords:** Volumetric data, Isosurfaces, Topology, Equally spaced, Similarity map

---

## 1. Introduction

Numerical simulations are extensively used by scientists to observe various phenomena that are not easily captured by real experiments. These simulations normally produce an ample amount of data, requiring effective tools to visualize and analyze them. A typical visualization presents the simulation results as a series of volumes. One of the essential techniques to gain insights into these volumes is isosurface rendering. To describe the structure of a volume, one can extract and visualize isosurfaces. These surfaces describe surface geometries with all points sharing the same isovalue. For insightful visualization, it is critical to select a set of salient isosurfaces that captures different features and characteristics of the underlying volume.

One common solution is to select a set of distinctive or representative isosurfaces from sample ones based on a certain similarity measure. For example, Tenginakai et al. [1] measured the similarities between isosurfaces using data histograms and higher order moments. Bruckner and Möller [2] derived distance fields from the sample isosurfaces and utilized mutual information to evaluate the similarity between the distance fields. The similarity values are organized in a matrix form named *isosurface similarity map* from which the representative isosurfaces are selected.

One major challenge exists for these approaches: it is essential for them to start with a set of reasonably good sample isosurfaces that capture different features in a balanced way. Otherwise, the features missing in the samples will not be recovered in the later stages, or the selection may be biased by favoring the features corresponding to more samples. However, straightforward sampling techniques do not guarantee the desired set of samples. Uniform sampling is likely to miss some

features when many of them reside in a small value range. Although sampling according to histograms of voxel values, i.e., placing more samples in the value ranges with more voxels, may alleviate this problem to some degree, it still suffers from oversampling as the value ranges with more voxels do not necessarily indicate more distinctive features.

Another key challenge is posed by the scale and complexity of the data generated by numerical simulations. To obtain a comprehensive understanding of physical phenomena, the simulations usually involve multiple variables and their interactions over time, resulting in large-scale time-varying multivariate volume data sets. This requires a surface-based analysis to be efficient in two aspects. First, the distance between two isosurfaces should be measured efficiently. Second, the number of distance calculations should be minimized so that one can afford to take a full run and draw a complete picture of the data. Previous approaches focused more on the definitions of similarity measures and less on performance optimization. For example, it took around 25 minutes to process a single volume with the isosurface similarity map approach [2]. This cost becomes prohibitive when analyzing a typical time-varying multivariate data set with tens of variables and hundreds of time steps.

In this paper, we present an approach for identifying nearly equally spaced isosurfaces, so that the distance between neighboring surfaces is as similar as possible to the average distance. In flow visualization, creating evenly spaced or mutually distant streamlines or stream surfaces has been well studied [3, 4, 5]. However, to the best of our knowledge, creating equally or evenly spaced isosurfaces has not been investigated. Our solution ensures that the isosurfaces corresponding to neighboring isovalues are distinct enough according to the given distance measure. When identifying a small number of isosurfaces, we

can consider the resulting isosurfaces as salient features on their own. Compared to the similarity-based approaches for identifying representative isosurfaces, our approach does not require the isosurfaces to be selected from a limited set of sample ones. It not only has a wider search space but also explicitly controls over the resulting isosurfaces, which can potentially lead to better results. Compared to the topology-based approaches, our method is more flexible when equipped with different distance measures. Although this offers great flexibility, our method relies on features being a function of isovalues. Given that precondition, our algorithm may capture the topological changes if the distance measure is topology-aware, and it may produce isosurfaces with distinct shapes if the distance measure is shape-aware. In addition, when a large set of isosurfaces is identified, the results can serve as reliable input to other volume analysis and visualization tasks. In our experiment, we find that taking our results as the input, the representative isosurfaces selected using isosurface similarity map [2] and k-means [6] can be improved. The comparison results will be presented in Section 4.2.

Our approach includes two stages: an estimation stage that quickly converges to a rough solution within a few iterations, and a refinement stage that optimizes the estimation. For both stages, only the distances between neighboring isosurfaces are needed at each iteration. Leveraging the parallel computation of GPU, we can process each iteration efficiently. In addition, our approach can be flexibly customized with various distance measures to meet different needs. In our experiment, we compare the performance and sampling results using the isosurface similarity map (ISM) measure [2] and the mean of the closest point distances (MCP) [7].

The contributions of our work are as follows.

- First, we present a feasible solution to identifying nearly equally spaced isosurfaces, an important yet seldom investigated problem. We shall see that our solution does not fully converge in general but we are able to find a solution with acceptable quality and performance trade-off. Compared to similarity- or topology-based methods, the set of isosurfaces generated by our method provides an advantageous visual summarization of the volumetric data, especially when the number of surfaces is small.
- Second, we perform a thorough study to compare parameter choices, distance measures, and qualitative results, followed by a list of findings for other researchers to follow. The proposed solution can be adopted by others for incorporation into their high-performance volume data analysis and visualization workflow.

## 2. Related Work

To analyze and visualize volumetric data sets, researchers have sought different kinds of methods to understand the structures of volumes. The *distribution-based* methods focus on the distributions of certain properties of the volume and identify the

salient structure based on their corresponding statistical characteristics. The *topology-based* methods analyze the topological structure of the volumes and highlight the structures corresponding to topological changes. The *similarity-based* methods measure the similarity between volume representations such as isosurfaces and derive the representative ones based on their similarities.

**Distribution-based methods.** Understanding the relationships between the volume distribution and the isosurfaces allows us to identify salient features. For instance, Tenginakai et al. [1] detected salient isosurfaces using local higher order moments (LHOMs). LHOMs are computed and plotted for different sample values for a semi-automatic selection. Scheidegger et al. [8] applied Federer’s Coarea Formula to improve the isosurface statistics by weighting with the inverse gradient magnitude. Duffy et al. [9] developed a mathematical model for continuous functions and proved the convergence to continuous statistics for regular lattices. Pekar et al. [10] proposed to use Laplacian weighted histograms for significant isovalue detection. However, the distribution of a volume data set does not translate to the spatial relationship among surfaces extracted, which is the focus of this paper.

**Topology-based methods.** These methods extract structures that essentially characterize properties of space such as convergence, connectedness, and continuity, providing a concise description of the overall structure of a volume. Bajaj et al. [11] proposed the contour spectrum, an interface combining the contour tree together with a variety of isosurface statistics, such as area and enclosed volume. Bremer et al. [12] presented the cancellation tree for describing the simplification of a Morse-Smale complex. Each simplification step cancels a pair of critical points, i.e., minima and maxima. The cancellation tree encodes the simplification steps and provides the connections among critical points. They further extended this approach to the hierarchical merge tree, which is a tracking graph that describes the temporal evolution of features [13]. Carr et al. [14] proposed to use the contour tree to encode the nesting relationships among isosurfaces. It also serves as an interface that allows users to select contours for operations such as removal, evolution, and tracking. Correa et al. [15] introduced the topological spine that connects critical points along the steepest ascending or descending directions. In addition, it includes geometric and contour nesting information, providing better spatial reasoning.

Although rigorous, topology-based methods normally capture minute topological changes, which lead to a large number of isosurfaces for a volume with complex topological variations. This, however, may not always be necessary for users to understand the overall structure of the volume. In contrast, our approach generates a small set of nearly equally spaced isosurfaces which are more amenable for user observation: each surface is distinct enough and they are mutually distant in the space. Such a set of isosurfaces could also be useful as a visual summarization of the underlying volume.

**Similarity-based methods.** Recent works often seek to measure the similarities between a set of sample isosurfaces and derive the structure of the entire volume. For example, Bruck-

ner and Möller [2] evaluated the similarity between isosurfaces and organized them in the form of an isosurface similarity map. The similarity between two isosurfaces is defined as the mutual information shared by the distance fields of the two isosurfaces. Representative isosurfaces are identified using the isosurface similarity map, which stores all pairwise similarity values. Haidacher et al. [16] extended this approach to compare isosurfaces extracted from multiple volumes. Wei et al. [17] proposed a similarity measure between two isosurfaces based on intermediate level-set surfaces. The values on the intermediate surfaces are sampled from the volume and their entropy values are used to evaluate whether the level-set surfaces align well with the intermediate isosurfaces. Recently, Ma et al. [6] used a tensor-based perceptual distance measure that simulates the human visual system and employed k-means clustering to select representative isosurfaces for comparing different volumetric data compression approaches.

These methods, including our approach, require distance fields of isosurfaces to be computed. As a common need of many applications, accelerating the distance field computation has been extensively studied [18, 19, 20, 21, 22]. Yu et al. [22] presented the parallel distance tree that distributes the workload to multiple processors guided by a coarse global distance tree. Each processor then constructs a local distance tree and derives the distance fields. To compute the distance field, the bounding volume hierarchy (BVH)-tree is often used to identify the closest points. Liu and Kim [20] proposed the multi-BVH that combines the octree and BVH-tree. The use of octree provides additional information to reduce the number of BVHs to be traversed. Karras [23] introduced a GPU-based method to construct BVH-trees in parallel, which is by far the fastest GPU solution available. In Section 3.4, we will discuss how we use Karras’ algorithm to accelerate distance computation.

### 3. Our Approach

We propose a two-stage approach for finding nearly equally spaced isosurfaces. Both stages run over several iterations aiming for convergence. First of all, during the *estimation stage*, we measure the distance between every pair of neighboring isosurfaces and resample the isovalues based on these distances using linear interpolation. This stage, however, assumes piecewise linearity of the distance function between neighboring isosurfaces, which does not hold in general. In our experiment, it approaches approximate solutions in a few iterations but normally does not converge to the optimal solution. Therefore, we introduce the *refinement stage* that adopts a binary search strategy to adjust each isovalue so that its surface has equal distance to its two neighbors. By repeating this process for several iterations, we achieve nearly equal distances between all neighboring isosurfaces.

In this section, we denote a set of isovalues at iteration  $\tau$  as  $V^\tau = \{v_1^\tau, v_2^\tau, \dots, v_n^\tau\}$ , the isosurface corresponding to an isovalue  $v_i^\tau$  as  $s_i^\tau$ , and the distance between two values  $v_i^\tau$  and  $v_j^\tau$  as the distance between their respective surfaces,  $d_{i,j}^\tau$ , or more generally,  $d(v_i, v_j)$ .

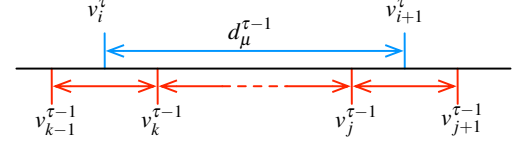


Figure 1: The estimation stage. Identifying isovalue  $v_{i+1}^\tau$  based on its left neighbor  $v_i^\tau$  and the distances evaluated at the previous time step  $\tau - 1$ .

#### 3.1. Estimation Stage

For the estimation stage, we start from a uniformly sampled set of isovalues  $V^0$  and gradually adapt the isovalues based on the previous set of isovalues and their distances. Specifically, at each iteration  $\tau$ , we first approximate the isosurface using the approximation scheme of Imre et al. [24] (refer to Section 3.4),  $s_i^{\tau-1}$  of each isovalue  $v_i^{\tau-1}$  at the previous iteration  $\tau - 1$  and evaluate the distance  $d_{i,i+1}^{\tau-1}$  between every pair of neighboring isovalues  $v_i^{\tau-1}$  and  $v_{i+1}^{\tau-1}$  using their corresponding approximated isosurfaces. The average distance  $d_\mu^{\tau-1} = (\sum_{i=1}^{n-1} d_{i,i+1}^{\tau-1}) / (n - 1)$  ( $\mu$  stands for the average) is considered to be the target distance to achieve at the current iteration  $\tau$ .

Then, starting from the first isovalue  $v_i^\tau = v_{\min}$ , where  $v_{\min}$  is the minimum isovalue, we resample each isovalue  $v_{i+1}^\tau$  that has approximately the target distance  $d_\mu^{\tau-1}$  to its left neighbor  $v_i^\tau$ , as illustrated in Figure 1. The distance is estimated under two assumptions. First, the distance between neighboring isovalues can be linearly interpolated. For example, the isovalue  $v_i^\tau$  falls between two previous isovalues  $v_{k-1}^{\tau-1}$  and  $v_k^{\tau-1}$ . We assume that the distance between  $v_i^\tau$  and  $v_k^{\tau-1}$  can be linearly interpolated using the distance  $d_{k-1,k}^{\tau-1}$ . Second, we assume that the distance can be added to estimate the distance between non-neighboring isovalues. For example, we assume that the distance  $d_{k,j}^{\tau-1}$  can be obtained using the summation of all neighboring distances between  $v_k^{\tau-1}$  and  $v_j^{\tau-1}$ , i.e.,  $d_{k,j}^{\tau-1} = d_{k,k+1}^{\tau-1} + \dots + d_{j-1,j}^{\tau-1}$ . In this way, we can iteratively identify the entire set of isovalues  $V_\tau$ . This process can also be considered as a parameterization based on the distances evaluated from previous neighboring isovalues.

We repeat this process for several iterations until a predefined minimum iteration number  $\delta_e$  is reached and the variation of neighboring distances stops decreasing. As shown in the first two rows of Figure 5, we can see that the estimation stage approaches the desired solution within a small number of iterations. Note that the computation of distances between neighbors, which is the most costly step, can be performed in parallel for each iteration. As previously mentioned, this stage is unlikely to converge since the two aforementioned assumptions do not hold for many volumetric data sets. In most cases, it is more likely to have  $d_{i,j} + d_{j,k} > d_{i,k}$  due to the triangle inequality. Therefore, the estimation stage only provides a rough solution, and an additional refinement stage is needed to obtain the optimal solution.

#### 3.2. Refinement Stage

In the refinement stage, we advocate a binary search strategy: placing the candidate isovalue in the middle of its two neighbors to identify an isosurface having the equal distance to its two neighboring isosurfaces. The distance function in this



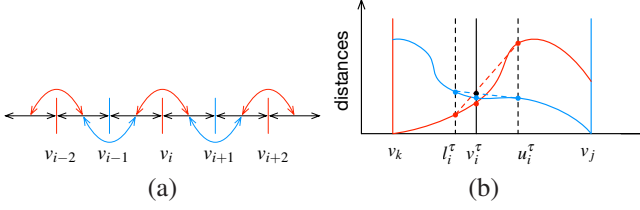


Figure 2: The refinement stage. (a) Adjusting odd-indexed and even-indexed isovalues alternatively. (b) Identifying an isovalue  $v_i$  that has the equal distance to its two neighbors  $v_k$  and  $v_j$ . The red (blue) curve in (b) represents the distance function from an isovalue in  $[v_k, v_j]$  to  $v_k$  ( $v_j$ ).

stage neither assumes linearity nor violates the triangle inequality. Unlike the estimation stage, this stage provides a slower but more robust process of convergence. This is achieved by adjusting the odd-indexed isovalues and even-indexed ones alternatively, as shown in Figure 2 (a). Specifically, the refinement stage is performed in multiple steps ( $\delta_r$ ). At odd steps, we adjust the red isovalues  $v_{i-2}, v_i, v_{i+2}$  with odd indices (assuming  $i$  is an odd number), so that they have the equal distance to their neighbor isovalues, i.e.,  $d_{i-3,i-2} = d_{i-2,i-1}$ ,  $d_{i-1,i} = d_{i,i+1}$ , and  $d_{i+1,i+2} = d_{i+2,i+3}$ , as indicated by the red arrows. Since the blue isovalues with even indices are fixed at odd steps, each odd-indexed isovalue can be adjusted independently in parallel. At even steps, we adjust the blue isovalues in the same fashion. Note that the blue and red arrows connect all distances between neighboring isovalues, which leads to equally spaced isovalues when this stage converges.

In every step, we use several iterations ( $\delta_\tau$ ) of a binary search strategy to identify an isovalue  $v_i$  that has the equal distance to its two neighbors  $v_{i-1}$  and  $v_{i+1}$ , as illustrated in Figure 2 (b). This means that the goal becomes finding one intersection point of the red and blue curves. At each iteration  $\tau$ , we maintain a lower bound  $l_i^\tau$  and an upper bound  $u_i^\tau$  that contain the intersection point between them. The lower and upper bound are initialized as  $v_k$  and  $v_j$ , respectively, i.e.,  $l_i^0 = v_k$  and  $u_i^0 = v_j$ . The lower bound maintains a property that it is always closer to  $v_k$  than  $v_j$ , i.e.,  $d_{k,l_i} < d_{l_i,j}$ , and the upper bound maintains a similar property in the opposite way, i.e.,  $d_{k,u_i} > d_{u_i,j}$ . Due to these properties, the red and blue curves must intersect somewhere in the middle as long as the distance functions are continuous.

At each iteration, we assume that the two distance functions change linearly between the bounds, as shown by the blue and red dashed lines in Figure 2 (b), and compute the intersection point, as indicated by the black dot. This intersection point provides the new isovalue  $v_i^\tau$  at step  $\tau$ , as shown by the black solid line in Figure 2 (b). We compute the distances  $d_{k,i}^\tau$  and  $d_{i,j}^\tau$  and determine whether  $v_i^\tau$  will replace the lower or upper bound. In this example, since  $d_{k,i}^\tau < d_{i,j}^\tau$ , indicated by the intersection points between the black solid line and the two curves, we replace the lower bound with  $v_i^\tau$ , i.e.,  $l_i^{\tau+1} = v_i^\tau$ , so that the properties of the lower and upper bounds still hold.

It is clear that the smaller the search range gets, the better the distance functions can be approximated by linear functions. As shown in Figure 2 (b), the curves between the bounds are

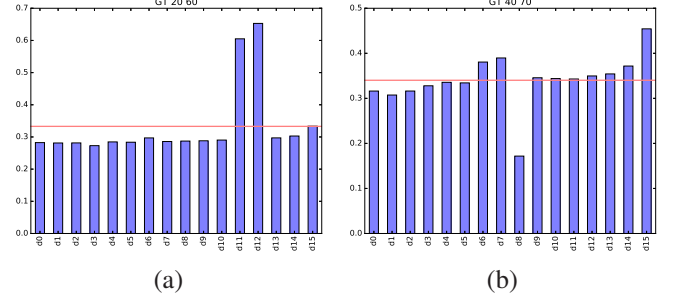


Figure 3: An example of neighboring distances from different time steps of the GT variable of the ionization data set. Every bar shows the distance  $d(i, i+1)$ . (a) shows one reason for a spike where two neighbors are too far apart. (b) shows a different cause, that is related to a jump discontinuity in the distance function.

nearly straight even if the distance functions themselves are not linear. This allows the desired isovalue to be identified within a small number of iterations. In our experiment, we find that five iterations per step is sufficient. For more details, please refer to Figure 5 and Section 4.1.

### 3.3. Convergence Stabilization

During the refinement stage, some proposed isovalues potentially lead to an adverse change of distance. This typically translates into distance values, named *spikes*, that are much higher/lower than the average distance, causing large average and maximum errors. Figure 3 demonstrates two common types of spikes: (1) the binary search successfully finds an isovalue that has mostly equal distances to its two neighbors, but these distances are larger than the average distance; and (2) the binary search fails to identify a desired isovalue, and the distance between this isovalue and one of its neighbors becomes a spike.

The first type of spikes between three isovalues  $v_{i-1}$ ,  $v_i$ , and  $v_{i+1}$  can be expressed as  $d_{i-1,i} \approx d_{i,i+1}$  and  $d_{i-1,i}, d_{i,i+1} \gg d_\mu$ . These spikes usually appear due to the underestimation of the differences of isosurfaces in the interval  $[v_{i-1}, v_{i+1}]$ . For example, in Figure 3 (a), after one step of the refinement stage, the binary search identifies an isovalue  $v_{12}$  whose distances to its neighbors (i.e.,  $d_{11,12}$  and  $d_{12,13}$ ) are about twice as high as the average distance, meaning that the interval between  $v_{11}$ ,  $v_{12}$ , and  $v_{13}$  may be too large. This type of spikes may gradually disappear since  $v_{i-1}$  and  $v_{i+1}$  will be moved closer to  $v_i$  in the next step of the refinement stage. For example, since  $d_{10,11}$  is much smaller than  $d_{11,12}$ ,  $v_{11}$  will be moved closer to  $v_{12}$  to reduce  $d_{11,12}$  for an equal distance between  $v_{10}$ ,  $v_{11}$ , and  $v_{12}$ . However, this type of spikes still causes a steep increase of the average and maximum errors, leading to an unstable status during the refinement stage.

To alleviate this problem, we propose a *spike treatment* that rejects isovalues leading to spikes. Formally, for every isovalue  $v_i^\tau$  that has been changed in step  $\tau$ , we compare  $d_{i-1,i}^\tau$  and  $d_{i,i+1}^\tau$  to  $d_\mu^\tau$  using their relative differences with respect to  $d_\mu^\tau$ .

If any of the two difference values surpasses a predefined spike threshold  $\delta_s$ , we reject  $v_i^\tau$  and replace it with  $v_i^{\tau-1}$ . Note that the old value  $v_i^{\tau-1}$  has more agreeable distances, as  $v_{i-1}$  and  $v_{i+1}$  are static in this step. Intuitively, by avoiding the steep

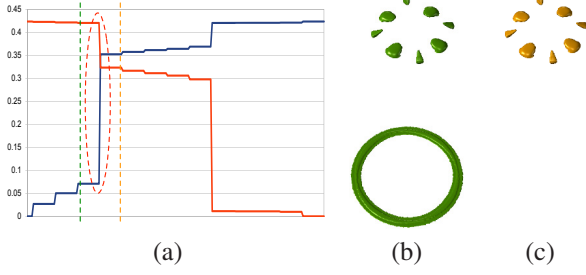


Figure 4: An example of discontinuous distance functions using the GT variable of the ionization data set. (a) shows the distances to isovalue 20547.8 (blue) and to isovalue 20680.4 (red). The horizontal axis represents sampled isovalues and the vertical axis represents the distance. (b) and (c) are the two isosurfaces corresponding to the isovalues highlighted by the dashed lines in (a).

changes, this treatment postpones, instead of preventing, the salient isovalue  $v_i$  to be discovered. Therefore, the entire refinement stage will exhibit a more smoothly convergence toward the best solution.

For the second type of spikes, the binary search fails to identify an isovalue  $v_i$  with equal distances to its neighbors. In this case, at least one of the distances  $d_{i-1,i}$  and  $d_{i,i+1}$  will differ from the average distance  $d_\mu$ . For example, in Figure 3 (b), the distance  $d_{8,9}$  is much smaller than the other distances between neighbors, therefore, leading to a larger error. This type of spikes is usually caused by a *jump discontinuity* in the distance functions between the neighboring isovalues.

A jump discontinuity (henceforth jump) appears when the distance function between two isovalues is discontinuous. An example is demonstrated in Figure 4. For the purpose of analysis, we consider two distance functions  $d_l(v_i)$  and  $d_r(v_i)$ , which map an isovalue  $v_i$  to its distances to its left and right neighbors, respectively. In this example, we densely sample 200 isovalues between two fixed isovalues 20547.8 and 20680.4, and compute the distance from each sampled isovalue to the fixed ones. In Figure 4 (a), the distances to isovalues 20547.8 and 20680.4 are plotted as blue and red lines, respectively. Unlike the case of two smooth distance functions, as demonstrated in Figure 2 (b), a steep change occurs, highlighted by the red dashed ellipse. This indicates that the two distance functions are not continuous at the corresponding isovalue.

Figure 4 (b) and (c) show the isosurfaces corresponding to the two isovalues on the two sides of this critical isovalue. We can see that this isovalue actually corresponds to the topological change with the lower ring emerging. The blue line appears below (above) the red line before (after) this change. This means, given the properties of the lower and upper bounds, the lower (upper) bound will always be on the left (right) of this isovalue. After several steps, the binary search will be trapped in a small value range centered at this critical isovalue. This does not only lead to a large error by itself but also stops the distance values from propagating from one side of the critical isovalue to the other side. Therefore, when a jump appears, we may only achieve two equal distances on the two sides of the critical isovalue.

To tackle this problem, we propose a *jump treatment* that first identifies the isovalue  $v^*$  of the jump and balances the dis-

tances on its two sides. By definition, a jump is a discontinuous point in the distance functions. Therefore,  $v^*$  can be detected through examining the following criterion

$$d(v_i, v^* + \epsilon) \gg d(v_i, v^*), \text{ and} \\ d(v^*, v_j) \gg d(v^* + \epsilon, v_j). \quad (1)$$

Instead of explicitly detecting  $v^*$ , we examine this criterion at each iteration of the refinement stage. Once a jump is encountered, we fix the upper and lower bounds of the binary search so that the jump will reside in the bounded interval, and push the isovalues from one side of the jump to the other side. Let  $V_l = \{v_k | 0 \leq k \leq i\}$  and  $V_r = \{v_k | j \leq k < n\}$  be the isovalue sets on the left and right sides of the jump, respectively, and let  $d_\mu(V_l)$  and  $d_\mu(V_r)$  be the average distances of the neighboring isovalues in  $V_l$  and  $V_r$ , respectively. Without loss of generality, assuming  $d_\mu(V_l) < d_\mu(V_r)$ , we take an isovalue from  $V_l$  and push it to  $V_r$ , so that the set of isovalues  $V_l$  becomes sparser and the set of isovalues  $V_r$  gets denser. This will lead to an increase of  $d_\mu(V_l)$  and a decrease of  $d_\mu(V_r)$ , thus achieving a better balance of the average distances on both sides.

### 3.4. Distance Measures

We experiment our approach with two different distance measures: the *mean of the closest point distances* (MCP) [7] and the *isosurface similarity map* (ISM) measure [2]. Other distance measures may be applied as well, according to the specific analysis goals.

**MCP distance.** The MCP distance between two isosurfaces  $s_i$  and  $s_j$  uses the Euclidean distance to compute the closest distance for every point  $p_k$  on  $s_i$  to any point  $p_l$  on  $s_j$  and vice versa. The MCP distance of  $s_i$  and  $s_j$  is defined as follows

$$d_{\text{MCP}}(s_i, s_j) = \frac{1}{2} (d(s_i, s_j) + d(s_j, s_i)), \text{ where} \\ d(s_i, s_j) = \frac{\sum_{p_k \in s_i} \min_{p_l \in s_j} \|p_k - p_l\|}{|S_i|}. \quad (2)$$

**ISM distance.** The ISM measure inspects the mutual information of the distance fields corresponding to two isosurfaces  $s_i$  and  $s_j$ . Based on the uniformly sampled distance fields of the two isosurfaces, a joint histogram can be computed to derive the mutual information. Again, we use the Euclidean distance to compute the distance fields for an isosurface  $s$ . For each grid point in the distance field, we record two closest distances from that point to the two isosurfaces  $s_i$  and  $s_j$ , and compute the joint histogram of the distances. The mutual information between two random variables  $X$  and  $Y$  can be computed from their joint histograms as follows

$$I(X, Y) = H(X) + H(Y) - H(X, Y), \text{ with} \\ H(X) = - \sum_{x \in X} p_X(x) \log(p_X(x)), \\ H(X, Y) = - \sum_{x \in X} \sum_{y \in Y} p_{X,Y}(x, y) \log(p_{X,Y}(x, y)), \quad (3)$$

where  $H(X)$  and  $H(Y)$  are the marginal entropies and  $H(X, Y)$  denotes the joint entropy of  $X$  and  $Y$ . In our case,  $X$  ( $Y$ ) is the distance from a grid point in the distance field to isosurface  $s_i$  ( $s_j$ ). We further normalize the mutual information

$$\hat{f}(X, Y) = \frac{2I(X, Y)}{H(X) + H(Y)}, \quad (4)$$

and convert the similarity measure into a distance measure

$$d_{\text{ISM}}(s_i, s_j) = 1 - \hat{f}(s_i, s_j). \quad (5)$$

**Approximation and acceleration.** Distance measures between isosurfaces often share two common steps: constructing isosurfaces and identifying the closest points of given points. When computing the ISM distance, the distance field of an isosurface requires the distance from each grid point to the closest point on the isosurface to be computed. When computing the MCP distance between two isosurfaces, for each point on one isosurface, the closest point on the other isosurface needs to be identified. We take three considerations from [24] to accelerate these two key steps.

First, we approximate each isosurface using a point set instead of extracting the actual surface. Generating the exact isosurface produces multiple points and their connections for each voxel. However, the connections are usually not involved in the distance computation and the points are often unnecessarily dense. This approximation scheme splits the volume into uniform blocks and examines each block to determine whether it contains the isosurface. The centers of blocks that contain the isosurface are considered as an approximation of the isosurface. Using this scheme, the error for computing the closest point is bounded by  $(\sqrt{3}/2)l$  (i.e., half the length of a block's diagonal  $l$ ).

Second, we build one BVH-tree for each isosurface to organize its approximation points. This allows the closest point on an isosurface to be queried efficiently. For construction efficiency, we use bounding boxes and leverage Karras' algorithm [23] to build each BVH-tree on GPU in parallel. The BVH-trees are stored in the graphics memory, so that multiple closest point queries can be performed in parallel. In addition, given a point, we estimate the upper bound of the distance to the closest point by uniformly sampling the approximation points. Since the approximation points are loosely ordered following the scanline order, this provides a tighter upper bound and therefore avoids many unnecessary branches of the BVH-tree to be traversed.

Third, both the distance fields and the approximation of isosurfaces can be downsampled to further reduce the time cost. The approximation can be downsampled by scanning blocks of voxels. The centers of blocks that contain the isosurface become the approximation in this case. The error of the closest point is bounded by  $(\sqrt{3}/2)l$ , where  $l$  is the edge length of a block in voxels. For the distance fields, it has been shown that the resolution can be reduced by eight folds along each dimension without sacrificing the quality of the resulting ISMs [2].

Through this acceleration measures, we can achieve a linear time complexity considering all steps to compute neighboring distances, except for building the BVH-trees. The initial approximation examines  $O(|V|)$  voxel, where  $|V|$  is the size of the volume. For the next step, building the BVH-tree, Karras reported the time complexity of  $O(n \log n)$  in the worst case [23]. Note that number of points from the approximation,  $n$ , is typically much smaller than  $|V|$ . Using the BVH-tree, querying the

closest point of a given point takes on average  $O(\log n)$  steps. The number of queries for this is bounded by either the size of the distance field ( $O(|V|)$ ) (when using ISM distance) or the size of another surface ( $O(n)$ ) (when using MCP distance). Using the GPU, multiple queries can be performed in parallel. For the ISM distance, we further compute mutual information, by examining every point in the distance field. In our experiment, we find that computing the distance between two isosurfaces already fully utilizes the computation power of a single GPU. Therefore, the cost of our approach is linear to the number of isosurfaces if a single GPU is used. Multiple GPUs, if available, can be readily utilized as computing multiple distances is embarrassingly parallel.

## 4. Results

We mainly run our experiments on a desktop with an Intel Core i7-4790 quad-core CPU @ 3.6 GHz, 32 GB RAM, and an NVIDIA GeForce GTX 760 GPU accelerator. For further exploration of time-varying data sets, we leveraged a cluster with a shared GPU queue. The shared GPU queue uses the following systems:

- 8 Quantum TXR231-1000R servers with dual Intel Xeon 12-core CPU E5-2650 v4 @ 2.20GHz, 128 GB RAM, and 4 NVIDIA TITAN X (Pascal) GPU accelerators;
- 8 Quantum TXR231-1000R servers with dual Intel Xeon 12-core CPU E5-2650 v4 @ 2.20GHz, 128 GB RAM, and 4 NVIDIA Tesla P100-PCIE-16GB GPU accelerators.

The queue distributes the workload on different machines depending on the availability. Since we were only interested in the number of iteration needed to achieve a good solution, we did not restrict our runs to a single hardware configuration. In the following, we first analyze our general approach quantitatively (Section 4.1) and qualitatively (Section 4.2), and then study the impact of the spike treatment and jump treatment (Section 4.3).

### 4.1. Quantitative Study

**Quality measures.** We evaluate the quality of a set of selected isovalues  $V = \{v_1, \dots, v_n\}$  based on the distances among neighbors (i.e.,  $d_{1,2}, \dots, d_{n-1,n}$ ) and the average distance  $d_\mu$ . For each distance  $d_{i-1,i}$ , we compute an error term  $e_{i-1,i}$  to indicate the difference between this distance and the average distance

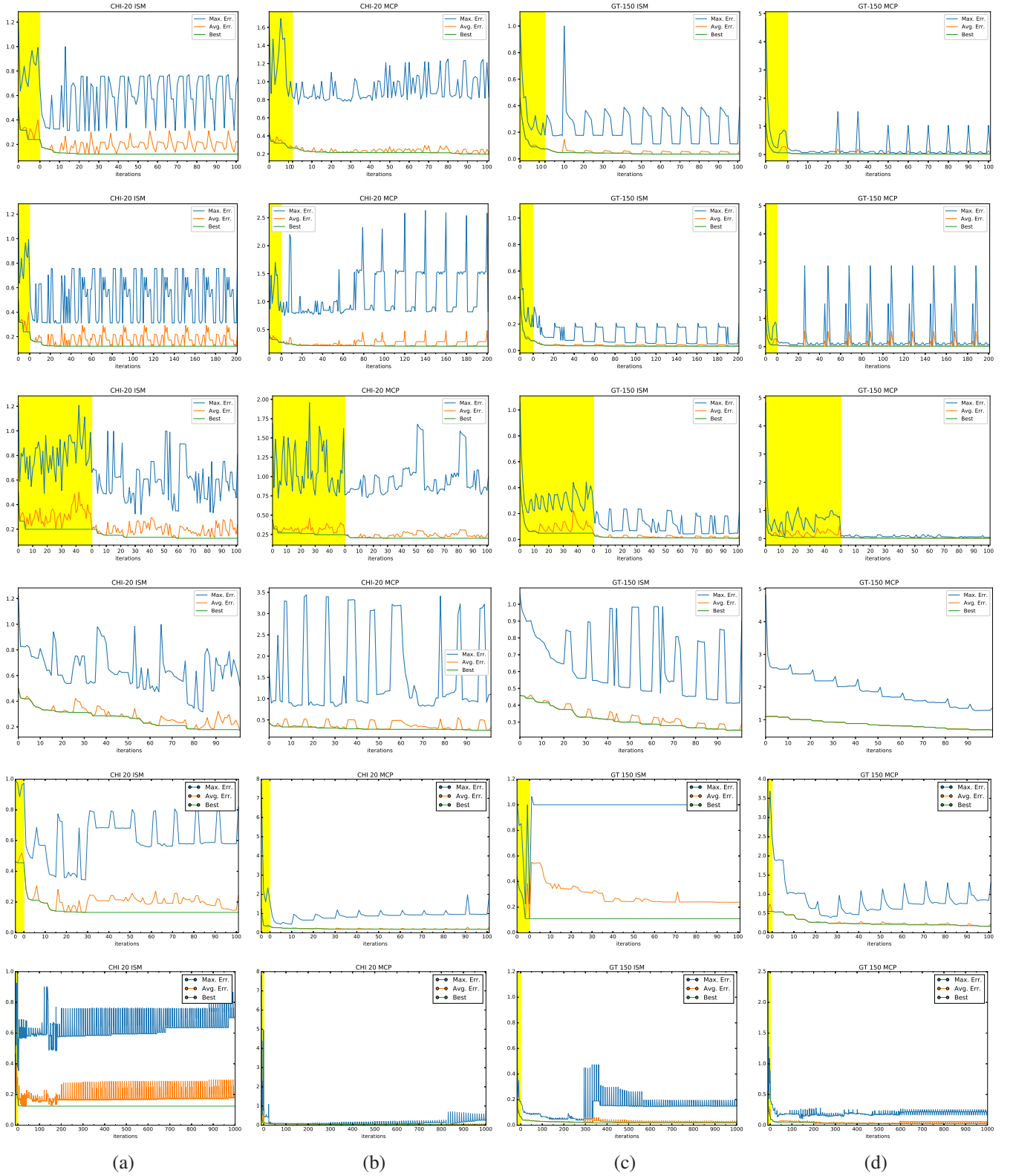
$$e_{i-1,i} = \frac{\|d_{i-1,i} - d_\mu\|}{d_\mu}, \quad (6)$$

where dividing the absolute difference by the average distance normalizes the error term. In this paper, we quantify the quality of selected isovalues using the average error

$$e_\mu = \frac{\sum_{i=2}^n e_{i-1,i}}{n-1}, \quad (7)$$

and the maximum error

$$e_{\max} = \max_{2 \leq i \leq n} \{e_{i-1,i}\}. \quad (8)$$





data set	dimension x, y, z, v, t	avg. # iterations		timing (sec.)		average error			refine improve	difference to best (%)			
		estimate	refine	estimate	refine	initial	estimate	best		20 iter.	40 iter.	60 iter.	80 iter.
atmosphere	147, 129, 49, 4, 121	11.1	57.9	3.36	27.61	0.437	0.153	0.081	47.29%	26.62	13.47	5.71	1.71
climate	360, 66, 27, 2, 120	11.8	47.5	2.87	22.21	1.060	0.251	0.165	34.12%	10.12	6.37	1.32	0.07
combustion	480, 720, 120, 5, 122	11.9	68.8	27.90	211.35	0.417	0.184	0.099	46.20%	25.39	13.16	5.31	1.16
hurricane	500, 500, 100, 11, 48	11.2	59.2	28.07	230.11	0.427	0.150	0.073	51.07%	40.50	18.08	7.65	3.92
ionization	600, 248, 248, 8, 199	11.5	63.9	20.70	164.63	0.597	0.287	0.188	34.33%	30.12	20.49	10.85	2.79
vortex	128, 128, 128, 1, 90	13.3	95.0	8.49	60.73	0.158	0.041	0.011	73.81%	41.42	28.95	20.22	15.27

(a)

data set	dimension x, y, z, v, t	avg. # iterations		timing (sec.)		average error			refine improve	difference to best (%)			
		estimate	refine	estimate	refine	initial	estimate	best		20 iter.	40 iter.	60 iter.	80 iter.
atmosphere	147, 129, 49, 4, 121	11.1	59.6	4.10	31.70	0.525	0.198	0.099	49.94%	57.60	19.91	8.92	2.74
climate	360, 66, 27, 2, 120	12.3	93.8	6.34	48.29	0.863	0.260	0.033	87.15%	206.16	104.33	53.49	27.29
combustion	480, 720, 120, 5, 122	12.9	72.2	91.51	646.55	0.560	0.205	0.099	51.54%	46.03	27.18	16.89	9.50
hurricane	500, 500, 100, 11, 48	12.8	88.8	49.57	379.31	0.281	0.061	0.010	82.17%	98.95	49.67	28.76	11.92
ionization	600, 248, 248, 8, 199	11.8	62.3	38.65	290.46	0.524	0.209	0.135	34.98%	27.79	15.26	6.08	2.72
vortex	128, 128, 128, 1, 90	11.0	80.0	15.42	126.15	0.533	0.194	0.072	62.81%	61.68	46.20	28.25	14.36

(b)

Table 1: Performances using (a) the ISM distance and (b) the MCP distance. The three columns of average errors show the initial average error of uniform sampling, the average error after the estimation stage, and the average error of the best solution. The column “refine improve” shows the percentage of average error reduced by the refinement stage. The four columns of “difference to best” show the percentage of difference between the average error of the best solution and the average errors after 20, 40, 60, and 80 iterations in the refinement stage.

Since the maximum error is usually determined by the nature of the data sets and the distance measures, as will be shown in Section 4.3, we focus on the average error and use it to determine the best solution, i.e., the set of isovalues with the minimum average error. We do not use the variation or standard deviation to evaluate whether the distances are similar since both of them are dominated by the maximum error when the other errors are small.

**Parameter choices.** Our approach has three parameters:  $\delta_e$  the minimum number of iterations in the estimation stage,  $\delta_\tau$  the number of iterations at each step in the refinement stage, and  $\delta_r$  the number of steps in the refinement stage. For simplicity, we use a 3-tuple  $\langle \delta_e, \delta_\tau, \delta_r \rangle$  to denote a parameter setting. Figure 5 shows the results of using two variables of the combustion and ionization data sets for both the ISM and MCP distance measures with six different sets of parameter values ( $\langle \delta_e, \delta_\tau, \delta_r \rangle = \langle 10, 5, 20 \rangle$ ,  $\langle 10, 10, 20 \rangle$ ,  $\langle 50, 5, 20 \rangle$ ,  $\langle 0, 5, 20 \rangle$ ,  $\langle 1, 5, 20 \rangle$ , and  $\langle 10, 5, 200 \rangle$ ). For each run, we plot the maximum error, the average error, and the current best solution over iterations. The current best solution is the one with the minimum average error obtained up to the current iteration.

We first investigate the impact of parameter  $\delta_\tau$ . In the top two rows of Figure 5, we fix the two parameters  $\delta_e = 10$  and  $\delta_r = 20$  and compare the performance of  $\delta_\tau = 5$  (first row) and  $\delta_\tau = 10$  (second row). At each step, having more iterations may potentially allow better convergence of the binary searches. But overall, we do not see a noticeable improvement of accuracy using  $\delta_\tau = 10$  over  $\delta_\tau = 5$  since the shape of the green curves (best solution) in the same column are mostly the same. With the same number of steps ( $\delta_r = 20$ ), this indicates that we obtain similar results using  $\delta_\tau = 10$  but with twice the number of iterations as using  $\delta_\tau = 5$ .

Next, we study the impact of parameter  $\delta_e$ . In the first and third rows of Figure 5, we use  $\delta_e = 10$  and  $\delta_e = 50$ , respectively. The other two parameters are fixed ( $\delta_\tau = 5$  and  $\delta_r = 20$ ). We find that more than ten iterations in the estimation stage are usually unnecessary since the best solution is mostly unchanged after ten iterations, as shown in the third row of Figure 5. In

addition, we do not find that having more iterations in the estimation stage helps the refinement stage reach the best solution faster. The green curves in the white background, corresponding to the best solution in the refinement stage, demonstrate similar decreasing patterns.

However, we still find that the estimation stage is necessary for the refinement stage to quickly reach its best solution. In the fourth row of Figure 5, we experiment our approach with only the refinement stage, i.e.,  $\delta_e = 0$ . With this setting, we find that the refinement stage approaches the optimal solution much slower. For example, using CHI of the combustion data set and the ISM distance measure shown in (a), the best solution slowly improves over the 100 iterations without the estimation stage and reaches the minimum average error of 0.179 at the last iteration. With the estimation stage, the best solution until the 20-th iteration in the refinement stage has an average error of 0.121, which is already very close to the minimum average error of 0.120 for the entire 100 iterations. The computation time of each iteration in the estimation and refinement stages is similar, since both of them are dominated by the computation of distances between neighbors. Therefore, including the estimation stage clearly gives a better performance.

The fifth row shows the results with only one iteration in the estimation stage, i.e.,  $\delta_e = 1$ . In contrast to setting  $\delta_e = 10$ , letting  $\delta_e = 1$  leads to a more stable convergence, implying that the algorithm got stuck in a local optimum. Note that for the CHI variable at time step 20, we sometimes obtain empty isosurfaces. In this case, we run the estimation stage for more iterations until we obtain a set without empty surfaces.

In the last row, we show the results with  $\langle \delta_e, \delta_\tau, \delta_r \rangle = \langle 10, 5, 200 \rangle$  (1000 total iterations in the refinement stage). For some data sets, we achieve a slightly better solution several hundreds iterations later than the best solution achieved within 100 iterations. However, the overall convergence pattern does not change. We believe that, given the time-quality trade-off, a relatively good solution can be found within 100 iterations.

In Figure 6, we show the visual differences among three sets of isosurfaces identified in 20, 100, and 1000 iterations, respectively, using the GT variable of the ionization data set at time



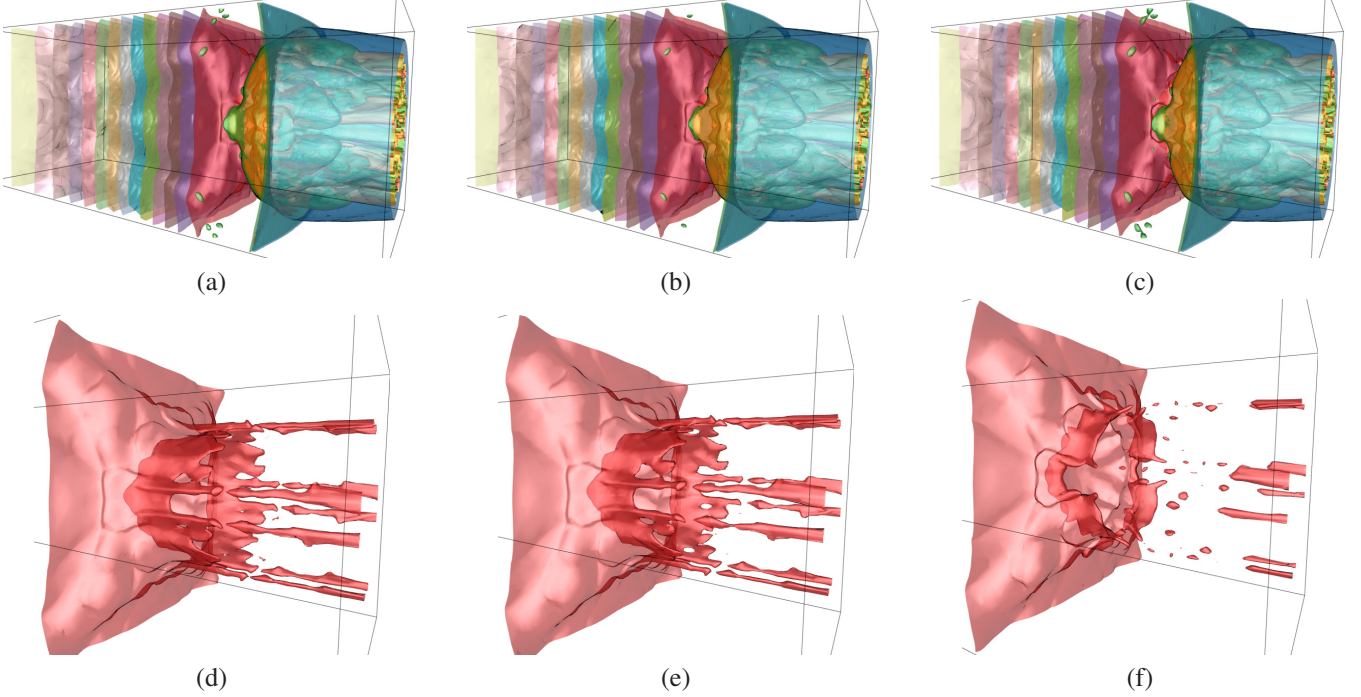


Figure 6: Comparison of the isosurfaces identified in different numbers of iterations using the GT variable of the ionization data set at time step 150. The isosurfaces are selected from the best solutions after 20 ((a) and (d)), 100 ((b) and (e)), and 1000 ((c) and (f)) iterations in the refinement stage. The top row shows 17 selected isosurfaces rendered together. The bottom row shows a single surface highlighting the fine differences.

step 150. While the best solution after 1000 iterations is found at iteration 333 with an average normalized error of 1.86%, the earlier ones 2.89% (at iteration 92) and 3.92% (at iteration 20) show small errors as well. Although the relative difference in average error seems enormous, their absolute difference is still small. Visual comparison confirms that similar isosurfaces are identified. In the top row of Figure 6, we show all isosurfaces corresponding to solutions at iteration 20 (a), 92 (b), and 333 (c), respectively. We find that the overall difference is barely visible. Therefore, to inspect more closely, we depict the fourth isosurface of the selected sets in the second row. We can clearly see a difference between the first two images ((d) and (e)) and the last one (f). Furthermore, subtle differences between (d) and (e) can be seen. Although single surfaces differ among the different solution sets, the overall sets look fairly similar, offering a comprehensive overview of the volumetric data set.

In our experiment, we use a large enough value of  $\delta_r = 20$  to study how the best solution evolves over iterations. We find that the setting of  $\langle \delta_e, \delta_r, \delta_r \rangle = \langle 10, 5, 20 \rangle$  usually yields good results in terms of timing and error. Therefore, we use this setting for reporting the remaining results.

**Timing and quality.** As shown in Table 1, we study the timing and quality performance using six data sets with different characteristics. For each data set, we use all the associated variables with three time steps selected (the beginning, middle, and ending time steps). Collected for each variable and each selected time step, the results are averaged for each data set.

Table 1 (a) shows the results using the ISM distance. Our approach produces mostly equally spaced isosurfaces with the average errors smaller than 0.1 for most of the data sets except

the climate (0.165) and ionization (0.188) data sets. This may be related to the structures of the data sets as their initial errors are the largest among these six data sets. The number of iterations in the estimation stage is stable for all data sets and slightly above our minimum number ( $\delta_e = 10$ ). In the refinement stage, our approach reaches the best solution around 60 iterations for most of the data sets, except the vortex data set (averaging 95 iterations). The climate data set even reaches solutions that are close to the best solution within 20 iterations, with only 10.12% difference. The other data sets except the vortex data set have differences less than or around 20% within 40 iterations and less than or around 10% within 60 iterations. Although having the slowest convergence, the vortex data set achieves the smallest average error (0.011) among all data sets after the refinement stage. In Figure 9 (k) and (l), we find that the green curve declines slowly after 15 iterations in the refinement stage. The higher percentages shown in the table are probably due to the small average error. The average errors are smaller than 0.3 for all the data sets after the estimation stage, and the refinement stage further reduces the average errors by at least 30%, which confirms the necessity of the refinement stage. Our approach performs efficiently using the ISM distance. To process one volume, it takes around one minute for the atmosphere, climate, and vortex data sets, and less than five minutes for the other data sets.

Table 1 (b) shows the results using the MCP distance. In general, we find that it takes more iterations for the refinement stage to approach the best solutions using the MCP distance. Three data sets reach the best solutions after 80 iterations. Within 40 iterations, only two data sets obtain good

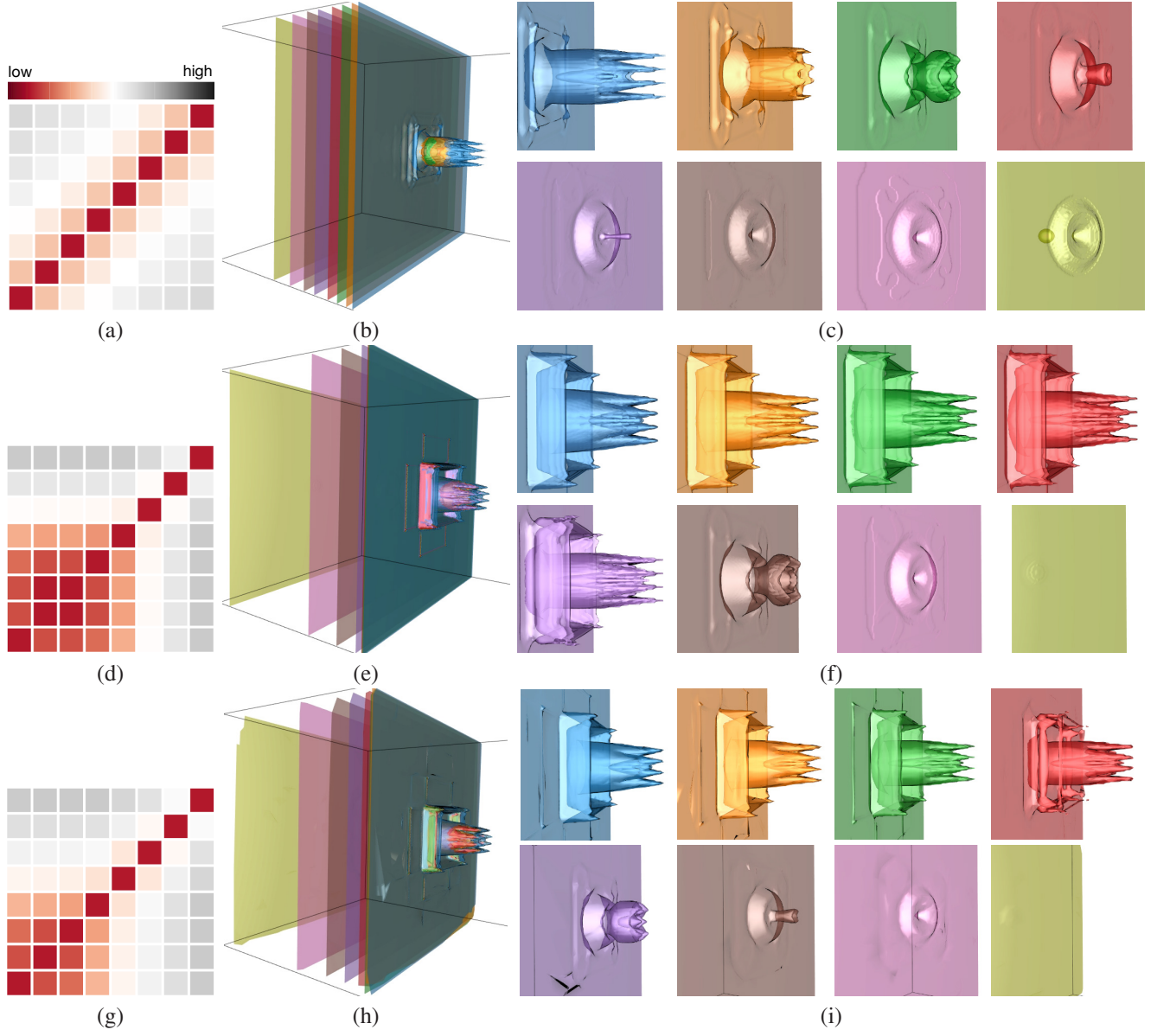


Figure 7: Comparison of our approach (top row), the ISM approach (middle row), and the k-means approach (bottom row) using the GT variable of the ionization data set at time step 10. (a), (d), and (g) show the distance matrices of the selected isosurfaces. Note that the distances are normalized and all matrices use the same color map. (b), (e), and (h) show all the selected isosurfaces in a single image. (c), (f), and (i) show the central regions of individual isosurfaces in separate images.

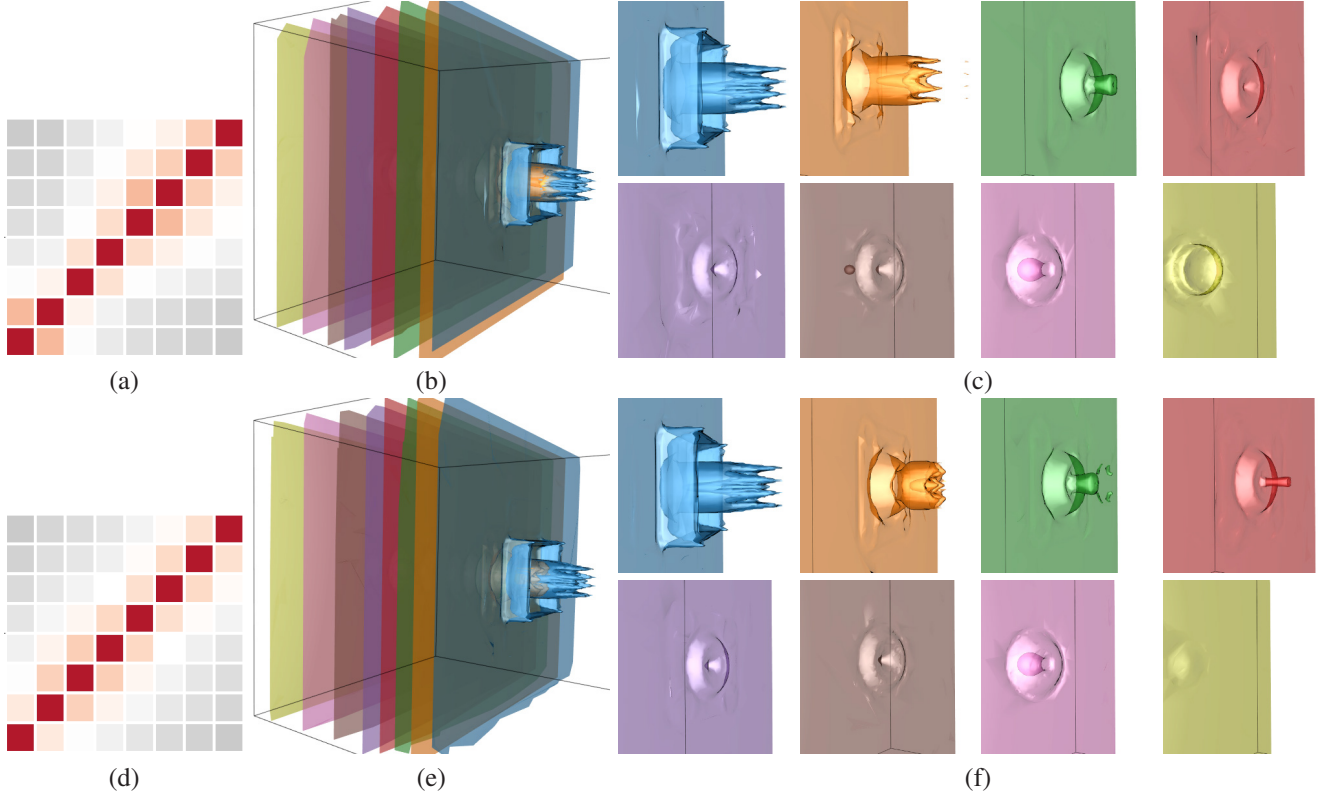


Figure 8: Comparison of results using a set of 128 nearly equally spaced isosurfaces as input for the ISM approach (top row) and the k-means approach (bottom row) using the GT variable of the ionization data set at time step 10. (a) and (d) show the distance matrices, (b) and (e) show the representative isosurfaces rendered together, while (c) and (f) show the individual surfaces.

solutions whose average errors have less than 10% differences from the best solutions. For five of the data sets, it takes 80 iterations to reach reasonably good solutions with less than 15% difference from the best solutions. However, we find that the average errors are usually smaller using the MCP distance. The largest average error is 0.135 using the ionization data set, and all the other data sets have average errors smaller than 0.1. The refinement stage provides more significant improvement using the MCP distance. It reduces the average error by at least 80% for two data sets and 49% for five data sets. Actually, we find that the data sets with a smaller best error usually benefit more from the refinement stage and take more iterations to reach the best solution. This is likely related to the intrinsic structures of the data sets. The distance functions are probably more continuous using these data sets so that the binary search in the refinement stage is less likely to be trapped by the discontinuous points in the distance function. Although the average error is smaller, we find that the MCP distance takes more time to compute. The combustion data set requires the longest total computation time of around nine minutes. The computation time for the other data sets varies from one to seven minutes.

#### 4.2. Qualitative Study

**Comparison to other approaches.** For visual comparison, we generate eight isosurfaces of the GT variable of the ionization data set at time step 10 using our approach, the ISM approach [2], and k-means clustering [6]. For our approach, we fix the minimum and maximum isovalues and compute eight

isovalues between them. For the other approaches, we evenly sample 128 isovalues and identify eight representative ones. We use Imre et al.'s [24] approximation to compute the ISM distances between the sampled isovalues for all three variants. The implementation of the ISM algorithm is based on the priority queue algorithm [2] and the k-means clustering is based on Lloyd's algorithm [25]. Figure 7 (a) shows the distance matrix of the isosurfaces selected by our approach. Note that in this, and the following images, the distance values are in  $[0, 1]$ . We can see that the cells recording the differences between neighboring isosurfaces (i.e., the cells that are next to the diagonal cells) share similar colors. This indicates similar distances between neighbors. Figure 7 (b) confirms this observation as the selected isosurfaces distribute evenly in the space. The eight isosurfaces demonstrate a smooth transition of the features at the center of each isosurface, as shown in Figure 7 (c).

In contrast, five of the representative isosurfaces selected by the ISM approach and four of the representative isosurfaces selected by the k-means approach are similar. In Figure 7 (d), we can see a  $5 \times 5$  block at the bottom left corner of the distance matrix of the representative isosurfaces, indicating high similarities among the corresponding isosurfaces. Similarly, Figure 7 (h) contains a  $4 \times 4$  block.

In Figure 7 (e), we can see that the five similar representative isosurfaces collapse in space. Therefore, five of the feature regions in the representative isosurfaces actually corresponds to the nearly identical structure, as shown in Figure 7 (f). Al-

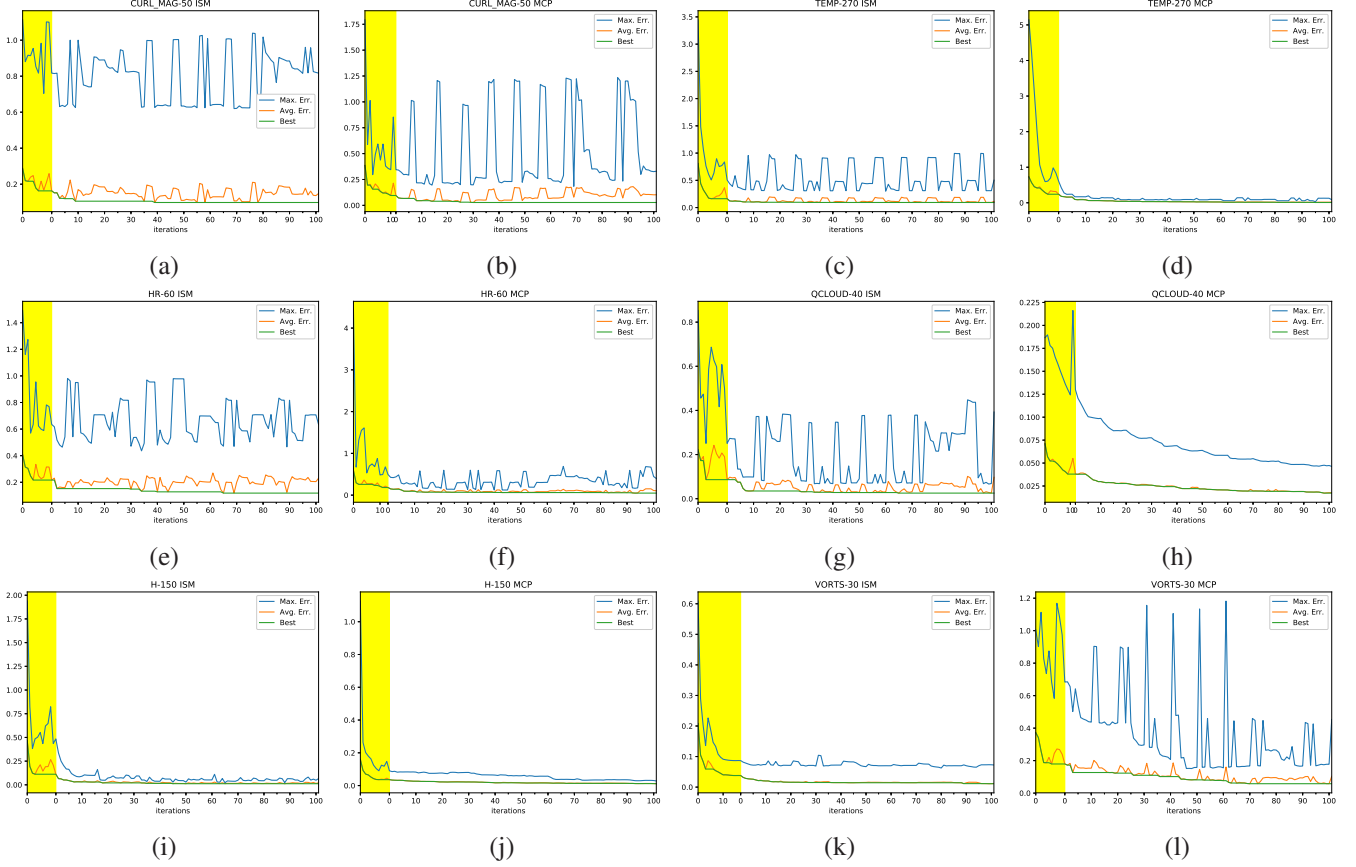


Figure 9: Typical error curves over iterations using the ISM distance (first and third column) and the MCP distance (second and fourth). (a) and (b) show the results of the curl magnitude (CURL\_MAG) variable of the atmosphere data set at time step 50. (c) and (d) show the results of the temperature (TEMP) variable of the climate data set at time step 270. (e) and (f) show the results of the heat release (HR) variable of the combustion data set at time step 60. (g) and (h) show the results of the cloud moisture mixing ratio (QCLOUD) variable of the hurricane data set at time step 40. (i) and (j) show the results of the H mass abundance (H) variable of the ionization data set at time step 150. (k) and (l) show the results of the vortex data set at time step 30.

though the ISM approach has a scheme to prevent similar isosurfaces to be selected [2], this scheme may be ineffective when the input is biased. As a matter of fact, more than half of the sampled isosurfaces in this volume correspond to the same structure. In general, we find that the distance matrix of the representative isosurfaces identified by the ISM approach often exhibits this kind of blocking effect for the structures captured by more sampled isosurfaces. This echoes that it is critical to producing unbiased isosurfaces as input for the surface-based volumetric data analysis algorithms. In addition, Figure 7 (h) shows the same effect for the k-means clustering with four similar isosurfaces selected. In Figure 7 (i), a closer inspection reveals that three of the four surfaces are very similar, with the fourth one being closely related to them.

In contrast, Figure 8 depicts the representative isosurfaces selected by the ISM [2] and k-means clustering [6] approaches using 128 nearly equally spaced isosurfaces as input instead of the uniformly sampled ones. We use the GT variable of the ionization data set at time step 10. In (a) and (d), the distance matrices do not show the strong blocking effect, which means that the problem of oversampling certain value ranges could be circumvented. Compared to Figure 7, we can see that there is a shift between the representative surfaces selected, allowing to

further explore previously overseen isosurfaces. For example, the third surface in the second row (pink) in both (c) and (f) has not been discovered previously. This further indicates that an unbiased input may improve the understanding of the underlying surfaces.

**Comparison of ISM and MCP distances.** For a qualitative study of the impact of distance measures, we first investigate the error curves using the ISM distance and the MCP distance, as shown in Figure 9. For each volume, we chose to identify 15 equally spaced isovalues between the minimum and maximum isovalues. In general, the curves confirm our finding in Table 1 that the MCP distance has a slightly smaller average error. The only exception is the vortex data set. Figure 9 (k) and (l) show that the MCP distance converges slower with unstable spikes of the maximum error curve for this data set. Figure 10 shows the distance matrix and a set of selected isosurfaces using the vortex data set for each distance measure. In Figure 10 (a), we can see that the distances between neighbors are actually similar for both measures. However, the two distance measures behave differently with this data set: the MCP distance changes in a smoother manner when the isovalues become more different, while the ISM distance seems to better distinguish isovalues in a smaller value range. For each distance measure, we evenly



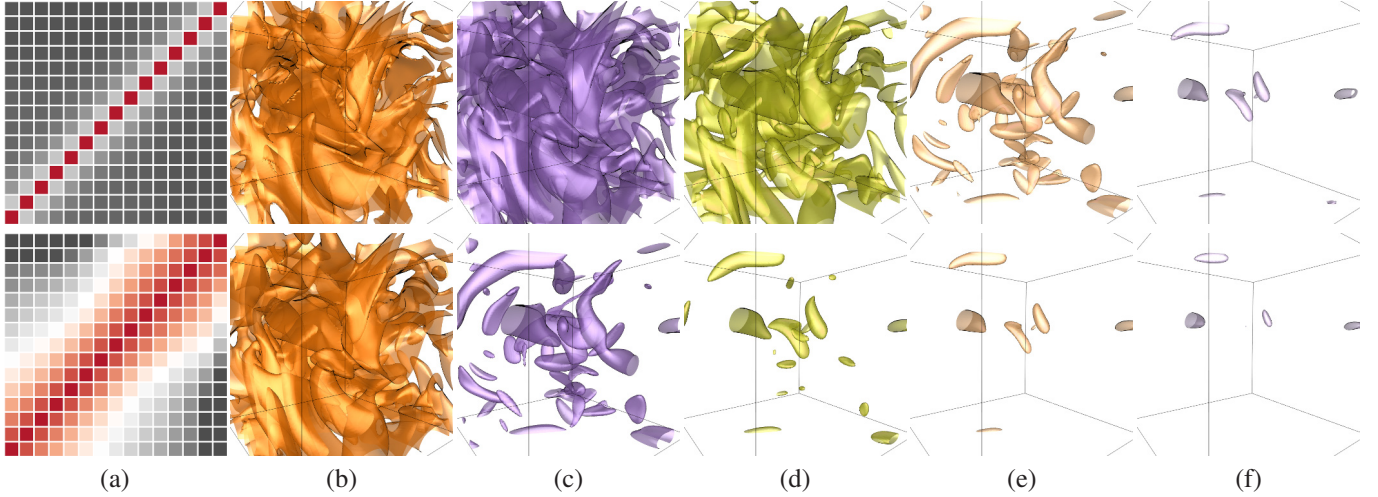


Figure 10: Comparison of the ISM distance (top row) and the MCP distance (bottom row) using the vortex data set at time step 30. (a) shows the distance matrices of the selected isosurfaces. (b) to (f) show the two sets of isosurfaces chosen by the respective distance measures.

select five isosurfaces (the second, fifth, eighth, 11th, and 14th) from the set of fifteen isosurfaces, as shown in Figure 10 (b) to (f). We find that the ISM distance identifies more large-scale isosurfaces while the MCP distance selects more small-scale isosurfaces. This is probably because the ISM distance better distinguishes the large isosurfaces and the MCP distance better differentiates the small ones using this data set.

We then investigate the combustion data set, for which the average errors are similar using both measures. We show the results of the HR variable in the top two rows of Figure 11. For this variable, although both the error curves in Figure 9 (e) and (f) and the distance matrices in Figure 11 (a) indicate smaller errors using the MCP distance, we find that the isosurfaces selected using the two distance measures are actually similar, as shown in the top two rows of Figure 11 (b) to (d). The isosurfaces are the ninth, 12th, and 14th from the fifteen selected ones. The first nine isosurfaces all demonstrate small-scale structures, which are visually similar. For the YOH variable, the distance matrices and the fourth, eighth, 12th isosurfaces (evenly sampled) are shown in the bottom two rows of Figure 11. We can see that the ISM distance selects more small-scale isosurfaces, which contradicts our findings with the vortex data set. We actually find that both measures are sensitive to changes on small-scale isosurfaces in general. This leads to the conclusion: the behavior of the two distance measures heavily depends on the spatial distribution of the isosurfaces. While large isosurfaces usually have stable spatial distributions, changes (even if they are tiny) on small isosurfaces may lead to significantly different spatial distributions. Therefore, due to the nature of these two measures, the differences among small isosurfaces are often emphasized.

#### 4.3. Discussion

**Time-varying data sets.** We further experiment possible solutions to improve the performance of time-varying data sets. In Section 4.1, we demonstrate that the refinement stage converges much faster with the use of the estimation stage, which

data set	# vol.	better vol. (%)		# iter. (%)	
		best	10% diff.	best	10% diff.
atmosphere	242	87.6	91.3	57.2	38.7
climate	151	72.9	99.3	91.4	35.20
combustion	61	0.0	1.6	N/A	18.93
hurricane	19	79.0	84.2	42.6	33.7
ionization	25	16.0	24.0	57.3	57.2
vortex	45	77.8	82.2	21.0	18.4

Table 2: Using the isovalues selected at the previous time step as initial isovalues for the next time step. “# vol.” shows the number of volumes experimented with. “better vol. (%)” shows the percentage of volumes achieving better results. “# iter. (%)” shows the percentage of iterations spent to achieve better results. “best” indicates that the result is better than the best solution obtained from the normal run, and “10% diff” indicates that the result is within 10% difference from the best solution of the normal run.

indicates the importance of a good initial set of isovalues. Observing that the structures of volumes usually change gradually over the time steps, we hypothesize that using the isovalues selected for the same variable at the previous time step will speed up the computation. Although this strategy has not been fully studied, we discuss some preliminary findings. We use a cluster with a shared GPU queue to experiment with the six data sets shown in Table 2. For each variable, we have a normal run that starts from the uniform sampling of the value range for each time step, and a run that starts from the isovalues selected at the previous time step. Each computation node in the GPU queue performs one run of a variable. Since the cluster contains computation nodes of different configurations, we compare the performance using the number of iterations instead of the computation time.

Our experiment shows that we do not always get better results by starting from the isovalues at the previous time step. Keeping the isovalues produces better results than the normal run for more than 70% of the volumes with four data sets. However, for the combustion and ionization data sets, this strategy fails to produce better or even similar results. For the volumes that better results are obtained, it generally takes much fewer it-

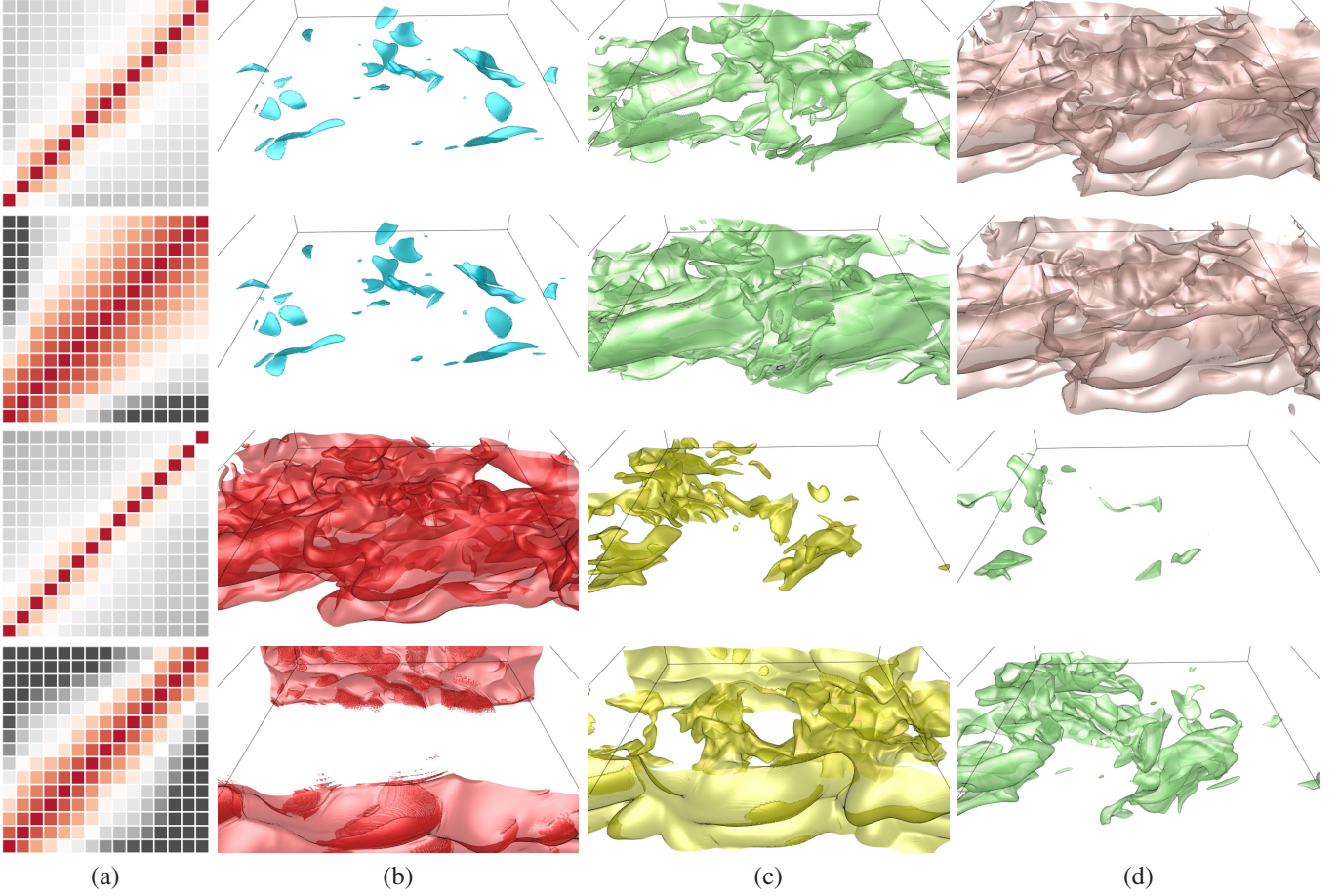


Figure 11: Comparison of the ISM distance and the MCP distance using the heat release (HR) and hydroxyl radical mass fraction (YOH) variables of the combustion data set at time step 60. Rows from top to bottom show the results using HR with ISM distance, HR with MCP distance, YOH with ISM distance, and YOH with MCP distance, respectively. (a) shows the distance matrices of the selected isosurfaces. (b) to (d) show the four sets of isosurfaces chosen by the respective combinations of variable and distance measure.

erations to achieve the better results (less than 60% for three of the four data sets). For the vortex data set, it only takes 21% of the number of iterations compared to the normal run. For five of the data sets, it takes less than 40% of the number of iterations to achieve a similar result. However, the conditions under which this strategy will perform effectively are still not clear. We further investigate the impact of the overlap percentage of the value ranges at neighboring time steps and the average error at the previous time step, but none of them exhibits a significant impact on the performance. It seems that the performance of this strategy heavily relies on the nature of the data since for all the data sets shown in Table 2, the percentage of volumes with better results is either higher than 70% or less than 20%. If we can determine in advance that a time-varying data set is suitable for this strategy, nearly  $2\times$  speedup can be obtained.

**Spike treatment.** To analyze the impact of the parameter  $\delta_s$ , we conduct an experiment using the common setting of  $\langle \delta_e, \delta_\tau, \delta_r \rangle = \langle 10, 5, 20 \rangle$  but varying  $\delta_s$  from 0.05 to 0.25 in steps of 0.05. Note that the original approach without spike treatment can be considered as setting  $\delta_s = \infty$ , meaning that all spikes are tolerated and will not be explicitly treated. Figure 12 shows the results for the GT variable of the ionization data set at time step 60. The figure represents typical error curves for

the ISM distance measure with different  $\delta_s$  settings. We can see that using a high tolerance value for  $\delta_s$ , as seen in (a) the original ( $\delta_s = \infty$ ), (e)  $\delta_s = 0.2$ , and (f)  $\delta_s = 0.25$ , yields high spikes in the maximum error, resulting in spikes in the average error. Dampening those instabilities by reducing the threshold translates to fewer negative changes as can be seen in (b)  $\delta_s = 0.05$ , (c)  $\delta_s = 0.1$ , and (d)  $\delta_s = 0.15$ . This further allows us to achieve a lower average error for the two variations shown in (c) and (d). Using  $\delta_s = 0.05$  may easily get trapped in a local optimum, since this parameter setting is too strict to allow any drastic changes that could resolve the problem. Setting  $\delta_s$  to 0.1 or 0.15 leads to the most stable convergence, showing that those values offer a good balance between allowing too little changes ( $\delta_s = 0.05$ ) and allowing too much changes ( $\delta_s \geq 0.2$ ). However, the best average errors achieved by setting  $\delta_s = 0.1, 0.15, 0.2, 0.25$  are similar. We will study the impact of  $\delta_s$  to the best average errors using more data sets later in this section.

**Jump treatment.** To analyze the impact of jumps and our treatment, we experiment our approach with several configurations (with and without spike treatment) using the GT variable of the ionization data set at time step 20. The error curves are shown in Figure 13. In (a), we can see the original without jump or spike treatments. The rest shows the different parameter set-

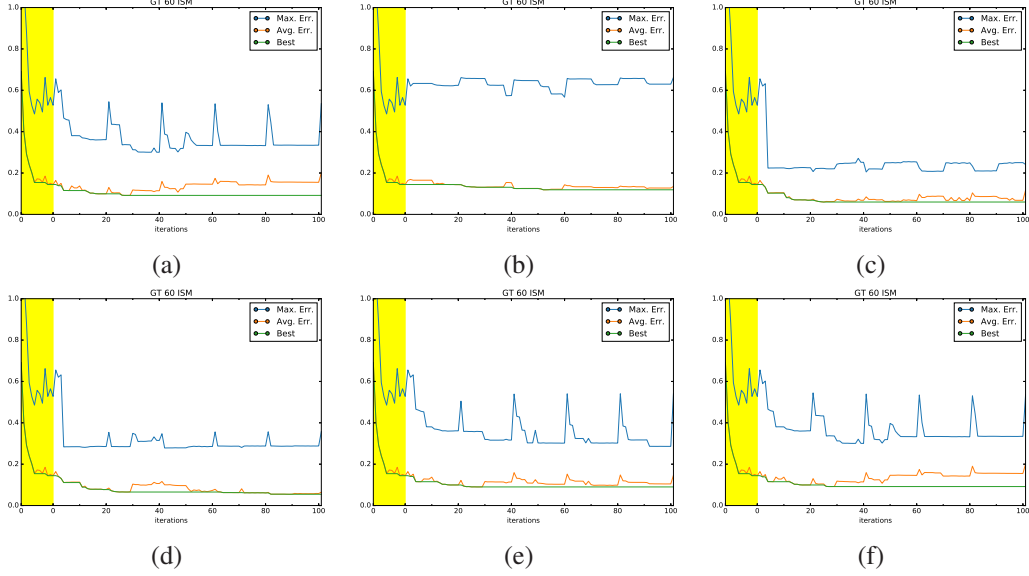


Figure 12: Comparison of the impact of  $\delta_s$  on the convergence and the final result using the ISM distance on the GT variable of the ionization data set at time step 60. (a) shows the original error curve over the set of iterations (i.e.,  $\delta_s = \infty$ ). (b) to (f) show the configurations with  $\delta_s = \{0.05, 0.1, 0.15, 0.2, 0.25\}$ , respectively.

tings for  $\delta_s$  to treat spikes while handling jumps at the same time. In our experiments, we encounter at most three jumps during the refinement stage using this data set. In (c) to (e), a strict choice of  $\delta_s = 0.05, 0.1, 0.15$  restricts isovalue changes to a degree where it is not possible to detect a single jump in the distance functions. By allowing more drastic changes to the isovalues ((f)  $\delta_s = 0.20$ , (g)  $\delta_s = 0.25$ , and (b)  $\delta_s = \infty$ ), we can see that jumps appear around strong fluctuation on the error curve. Treating these jumps helps to reduce the fluctuation from that point on. In (b), although the error curve still has strong spikes after treating the first jump, it finds a better solution than the original approach in a later iteration. An even stronger reduction in fluctuation of the error curve can be seen in (f). Dampened by spike treatment, the first jump is detected at iteration 70 and its treatment allows the curve to come down further, achieving the best solution for this data set at iteration 90. However, jump treatment does not always lead to better convergence or lower average and maximum error values. Figure 13 (g) showcases this. Instantly after the jump around iteration 50, the error function cannot recover, resulting in another jump quickly. This results in multiple fixed isovalues, allowing fewer changes in the set of isovalues at every iteration, which hinders the optimization process.

As mentioned previously, it is not always clear beforehand whether or not jump treatment improves the best solution due to multiple reasons. First, we treat each jump when it is detected at the refinement stage without the knowledge of all jumps. Therefore, treating one jump may lead to the appearance of an undiscovered jump. As shown in Figure 13 (g), the treatment may not be effective when multiple jumps are encountered. Second, pushing an isovalue over a jump does not guarantee that the distances on the two sides of the jump will be equal, especially when the jump occurs close to the end of the value range. In addition, this situation is often aggravated when additional jumps are encountered.

**Configuration selection.** As we have seen before, treating jumps and spikes can both have a positive or a negative impact on the achieved solution. In order to recommend an appropriate configuration, we run experiments among all time steps of different data sets and variables. Figure 14 shows the mean and standard errors of the best average errors with different configurations using four variables from four different data sets.

The top row shows the results collected using all the time steps. We can see that in most of the cases, one of our convergence stabilization configurations improves the overall solution or yields a solution similar to the original one. For (a), the CHI variable of the combustion data set, our experiment shows that treating the jumps but ignoring spikes outperforms the other methods by a huge margin. This is likely due to an initial set of isovalues being stuck in a local optimum and can only escape it by treating jumps while allowing huge spikes. For (b), the H variable of the ionization data set, we witness that a strict policy for spikes yields the best results. Interestingly, we can see that all configurations, except for  $\delta_s = 0.05, 0.1$  without jump treatment, and  $\delta_s = 0.2$  with jump treatment, have a very high standard error. This indicates that the structure of the volume differs heavily between time steps. In contrast, a loose spike treatment ( $\delta_s = 0.2, 0.25$ ) has a small standard error among all configurations using the SALT variable of the climate data set, as shown in (c). Similarly, in (d), the bar chart shows that too strict spike treatment can have a strong negative impact on the performance, using the VEL\_MAG variable of the atmosphere data set.

As there is no clear favorite among all data sets, we sample the first five time steps of a data set to see if we can predict a good configuration for the full run. These results are demonstrated in the bottom row of Figure 14. In (a) and (b), our method performs consistently over the time steps, showing the possibility to predict the best variation from the first couple of time steps. In (c), although we mispredict the best configura-



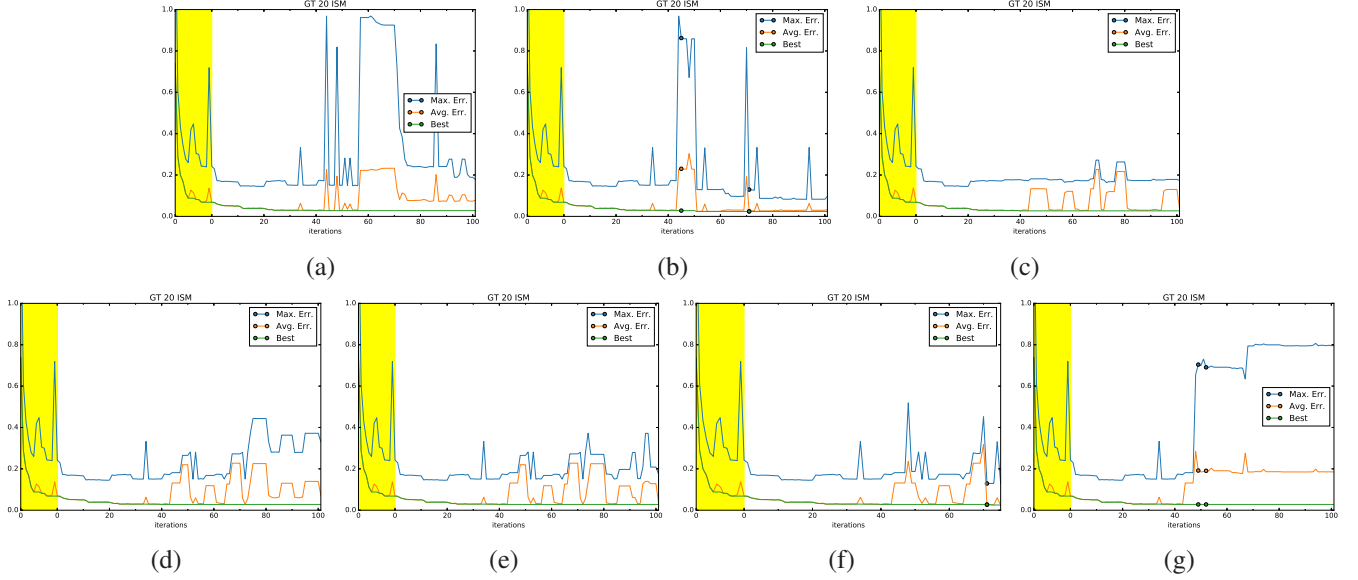


Figure 13: Error curves generated with different configurations using the GT variable of the ionization data set at time step 20. (a) shows the original approach without jump and spike treatments. (b) shows a configuration with only jump treatment. (c) to (g) show configurations with both jump and spike treatments while  $\delta_s$  increases from 0.05 to 0.25 in steps of 0.05. Dots on the curves indicate the iterations when jumps are detected.

tion, the predicted one still shows significant improvement over the original. However, in (d), the bar chart indicates that a bad configuration is recommended based on the results using the first five time steps. Although the recommended configuration ( $\delta_s = 0.15$  with jump treatment) performs similarly as the original approach, we fail to spot the best configuration ( $\delta_s = 0.15$  without jump treatment). In this example, the high standard errors among the first five time steps may indicate that the intermediate favorite is not a stable one for all the time steps. Overall, we cannot always predict the exact configuration that yields the best result for a given data set. However, when considering the first couple of time steps, we often identify a configuration of the convergence stabilization that outperforms the original.

## 5. Conclusions

We have presented a solution for identifying nearly equally spaced isosurfaces for volumetric data sets. Motivated by finding a small set of isosurfaces to better represent the underlying volume data in the spatial domain, we design a two-stage approach to seeking an approximated solution that maintains a good balance between quality and performance. The resulting surfaces are nearly equally spaced, and the user can freely choose the number of surfaces. Our study demonstrates the effectiveness of the proposed approach and leads to valuable feedback. To conclude, we summarize our key findings and major recommendations as follows.

First, our two-stage strategy is effective for achieving the best solution in a small number of iterations. Our experiment shows that both stages are necessary: without the estimation stage the refinement stage would require a lot more iterations to converge, and the estimation stage may never achieve a solution with a similar error as the refinement stage does.

Second, our approach can produce nearly equally spaced isosurfaces for most of the data sets, although some error may

be introduced by jump discontinuities in the distance functions. These points may divide the entire range of isovalues into multiple segments and prevent the isovalues from moving between neighboring segments, resulting in unequal distances among the segments. Our convergence stabilization scheme alleviates this situation by treating spikes and jumps explicitly, although the effectiveness depends on the specific data set.

Finally, our approach is independent of the choice of the distance measure. This provides great flexibility for users to apply a suitable distance measure according to their own needs. Our experiment performs effectively using both the ISM and MCP distance measures with a common parameter setting of  $\langle \delta_e, \delta_\tau, \delta_r \rangle = \langle 10, 5, 20 \rangle$  for all the data sets. To ensure smoother convergence without a strong restriction, we recommend using  $\delta_s = 0.2$  and ignoring jumps, as these settings either outperform or show similar results as the original across all data sets.

## Acknowledgements

This research was supported in part by the U.S. National Science Foundation through grants IIS-1456763, IIS-1455886, and CNS-1629914, and the NVIDIA GPU Grant Program. The authors would like to thank the anonymous reviewers for their insightful comments.

- [1] Tenginakai S, Lee J, Machiraju R. Salient iso-surface detection with model-independent statistical signatures. In: Proceedings of IEEE Visualization Conference. 2001, p. 231–8.
- [2] Bruckner S, Möller T. Isosurface similarity maps. Computer Graphics Forum 2010;29(3):773–82.
- [3] Jobard B, Lefer W. Creating evenly-spaced streamlines of arbitrary density. In: Visualization in Scientific Computing 1997: Proceedings of the Eurographics Workshop in Boulogne-sur-Mer France. 1997, p. 43–56.
- [4] Spencer B, Laramée RS, Chen G, Zhang E. Evenly spaced streamlines for surfaces: An image-based approach. Computer Graphics Forum 2009;28(6):1618–31.



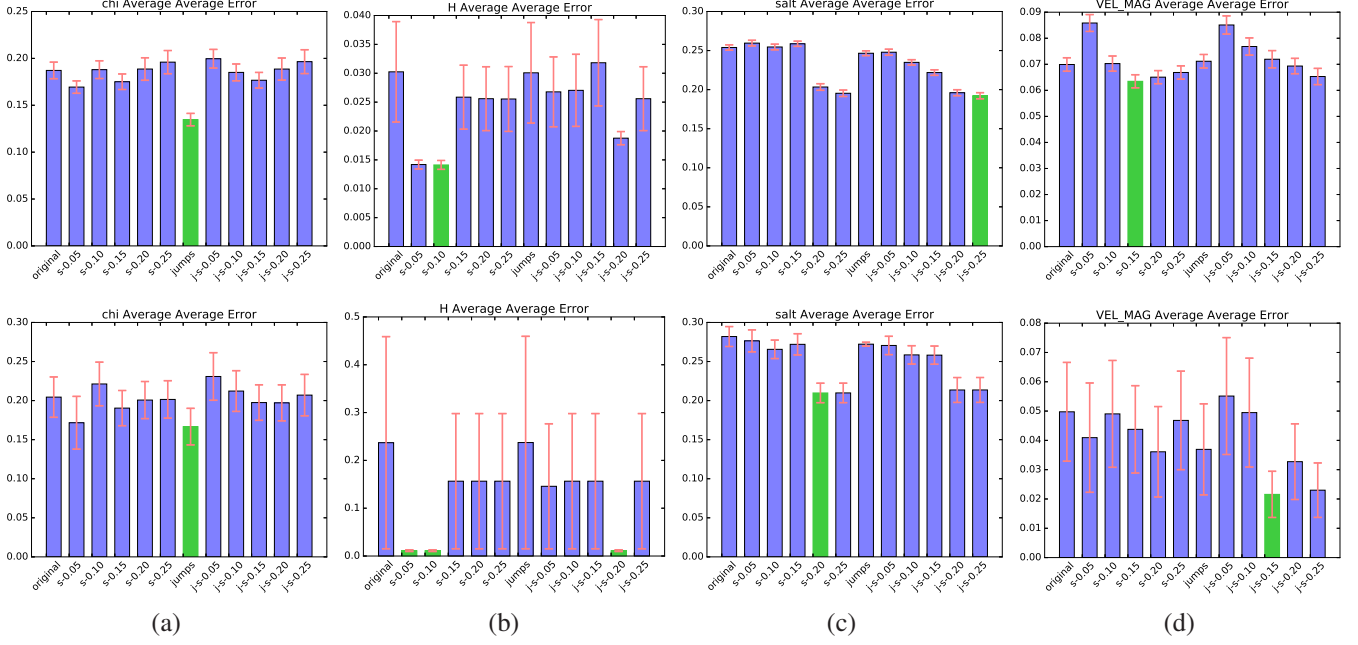


Figure 14: Comparison of the best average errors with different settings using (a) the CHI variable of the combustion data set, (b) the H variable of the ionization data set, (c) the salinity (SALT) variable of the climate data set, and (d) the velocity magnitude (VEL\_MAG) variable of the atmosphere data set. The top row shows the results using all the time steps and the bottom row shows the results using only the first five time steps. In each chart, a bar represents the mean of the average errors achieved over the respective time steps of the corresponding data set (lower is better). The bars from left to right correspond to the original ( $\delta_s = \infty$ ), spike treatment with  $\delta_s = 0.05, 0.1, 0.15, 0.2, 0.25$ , and jump and spike treatments with  $\delta_s = \infty, 0.05, 0.1, 0.15, 0.2, 0.25$ , respectively. The error bars in red indicate the standard error of the mean. The bars in green represent the best settings.

- [5] Schulze M, Martinez Esturo J, Günther T, Rössl C, Seidel HP, Weinkauff T, et al. Sets of globally optimal stream surfaces for flow visualization. *Computer Graphics Forum* 2014;33(3):1–10.
- [6] Ma B, Suter SK, Entezari A. Quality assessment of volume compression approaches using iso-value clustering. *Computers & Graphics* 2017;63:18–27.
- [7] Moberts B, Vilanova A, van Wijk JJ. Evaluation of fiber clustering methods for diffusion tensor imaging. In: *Proceedings of IEEE Visualization Conference*. 2005, p. 65–72.
- [8] Scheidegger CE, Schreiner JM, Duffy B, Carr H, Silva CT. Revisiting histograms and isosurface statistics. *IEEE Transactions on Visualization and Computer Graphics* 2008;14(6):1659–66.
- [9] Duffy B, Carr H, Möller T. Integrating isosurface statistics and histograms. *IEEE Transactions on Visualization and Computer Graphics* 2013;19(2):263–77.
- [10] Pekar V, Wiemker R, Hempel D. Fast detection of meaningful isosurfaces for volume data visualization. In: *Proceedings of IEEE Visualization Conference*. 2001, p. 223–30.
- [11] Bajaj CL, Pascucci V, Schikore DR. The contour spectrum. In: *Proceedings of IEEE Visualization Conference*. 1997, p. 167–73.
- [12] Bremer PT, Pascucci V, Hamann B. Maximizing adaptivity in hierarchical topological models. In: *Proceedings of International Conference on Shape Modeling and Applications*. 2005, p. 298–307.
- [13] Bremer PT, Weber G, Tierny J, Pascucci V, Day M, Bell J. Interactive exploration and analysis of large-scale simulations using topology-based data segmentation. *IEEE Transactions on Visualization and Computer Graphics* 2011;17(9):1307–24.
- [14] Carr H, Snoeyink J, van de Panne M. Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree. *Computational Geometry* 2010;43(1):42–58.
- [15] Correa CD, Lindström P, Bremer PT. Topological spines: A structure-preserving visual representation of scalar fields. *IEEE Transactions on Visualization and Computer Graphics* 2011;17(12):1842–51.
- [16] Haidacher M, Bruckner S, Gröller E. Volume analysis using multimodal surface similarity. *IEEE Transactions on Visualization and Computer Graphics* 2011;17(12):1969–78.
- [17] Wei TH, Lee TY, Shen HW. Evaluating isosurfaces with level-set-based information maps. *Computer Graphics Forum* 2013;32(3):1–10.
- [18] Krishnamurthy A, McMains S, Haller K. GPU-accelerated minimum distance and clearance queries. *IEEE Transactions on Visualization and Computer Graphics* 2011;17(6):729–42.
- [19] Larsen E, Gottschalk S, Lin MC, Manocha D. Fast distance queries with rectangular swept sphere volumes. In: *Proceedings of IEEE International Conference on Robotics and Automation*. 2000, p. 1–8.
- [20] Liu F, Kim YJ. Exact and adaptive signed distance fields computation for rigid and deformable models on GPUs. *IEEE Transactions on Visualization and Computer Graphics* 2014;20(5):714–25.
- [21] Seitzer J. Parallel computation of the Euclidean distance transform on a three-dimensional image array. *IEEE Transactions on Parallel and Distributed Systems* 2003;14(3):203–12.
- [22] Yu H, Xie J, Ma KL, Kolla H, Chen JH. Scalable parallel distance field construction for large-scale applications. *IEEE Transactions on Visualization and Computer Graphics* 2015;21(10):1187–200.
- [23] Karras T. Maximizing parallelism in the construction of BVHs, octrees, and k-d trees. In: *Proceedings of ACM SIGGRAPH/Eurographics Conference on High-Performance Graphics*. 2012, p. 33–7.
- [24] Imre M, Tao J, Wang C. Efficient GPU-accelerated computation of isosurface similarity maps. In: *Proceedings of IEEE Pacific Visualization Symposium*. 2017, p. 180–4.
- [25] Lloyd S. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 1982;28(2):129–37.