Volume 0 (1981), Number 0 pp. 1–11 COMPUTER GRAPHICS *forum*

Flow-based Temporal Selection for Interactive Volume Visualization

S. Frey and T. Ertl

University of Stuttgart, Visualization Research Center, Germany

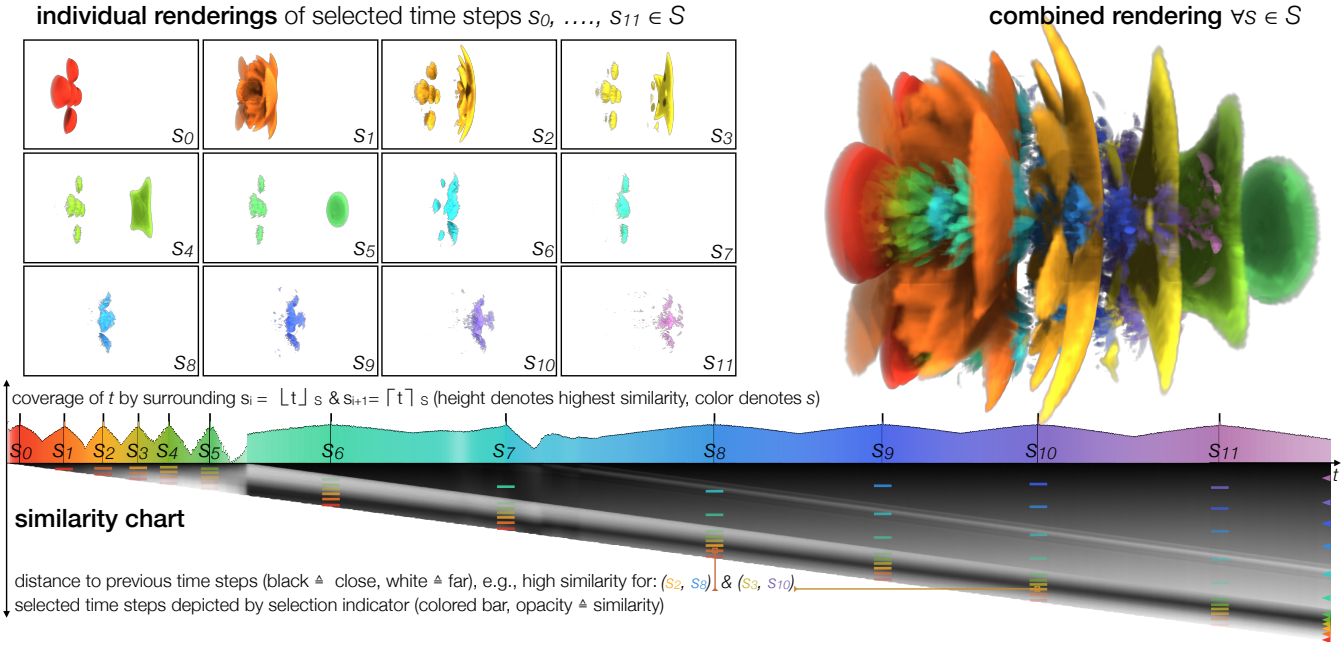


Figure 1: Selecting time steps $S = \{s_0, \ldots, s_{n-1}\} \subset T$ to optimally cover time series T based on flow-based distances. Here, five jets induce a wave moving from the left to the right ($s_0 - s_5$), with a similar movement occurring later at slower pace with lower mass ($s_6 - s_{11}$). Individual and combined renderings of time steps provide spatial relation and detail, and our similarity chart gives full coverage and distance information.

Abstract

We present an approach to adaptively select time steps from time-dependent volume data sets for an integrated and comprehensive visualization. This reduced set of time steps not only saves cost, but also allows to show both the spatial structure and temporal development in one combined rendering. Our selection optimizes the coverage of the complete data on the basis of a minimum-cost flow-based technique to determine meaningful distances between time steps. As both optimal solutions of the involved transport and selection problem are prohibitively expensive, we present new approaches that are significantly faster with only minor deviations. We further propose an adaptive scheme for the progressive incorporation of new time steps. An interactive volume raycaster produces an integrated rendering of the selected time steps, and their computed differences are visualized in a dedicated chart to provide additional temporal similarity information. We illustrate and discuss the utility of our approach by means of different data sets from measurements and simulation.

Categories and Subject Descriptors (according to ACM CCS): I.3.m [Computer Graphics]: Miscellaneous—

1. Introduction

Advances in parallel computing systems for simulations and high-accuracy measurement techniques drive the generation of time-dependent data sets with increasing resolution in both time and space. This data can feature millions of cells and thousands of time steps, and thus poses significant challenges for visual analysis. Even if all of the large data—potentially well-exceeding the available memory—could be presented to the user interactively, still issues due to occlusion and visual overload need to be prevented. While

relying on animation for time-dependent data is both a popular and natural choice, it has been shown to be ineffective as only a limited number of frames can be memorized by an observer (e.g., [JR05]).

To address these issues, our approach selects several time steps and integrates them into one visualization that adequately depicts the whole time range, conveying both spatial structure and temporal development. The adaptive selection of time steps for visualization is crucial, e.g., some processes of interest might occur in a relatively short time range, and thus a regular selection might miss it partly or even entirely. Furthermore, our selection process needs to be optimized for high performance to be able to handle large data sets. In the following, after a discussion of related work (Sec. 2), we exemplify our approach by means of an introductory example (Sec. 3a). We then present what we consider to be our main contributions:

- our integrated approach for temporal volume analysis that supports the adaptive progressive incorporation of new time steps (Sec. 3b,c). It consists of components for
- the flow-based distance computation between volumes via Delaunay-triangulated samples and minimum-cost flow (Sec. 4),
- the selection of a variable number of time steps such that the whole time series is optimally covered (Sec. 5),
- and the integrated rendering of selected time steps and visualization of respective similarity information (Sec. 6).

We finally present results in Sec. 7 and conclude our work in Sec. 8.

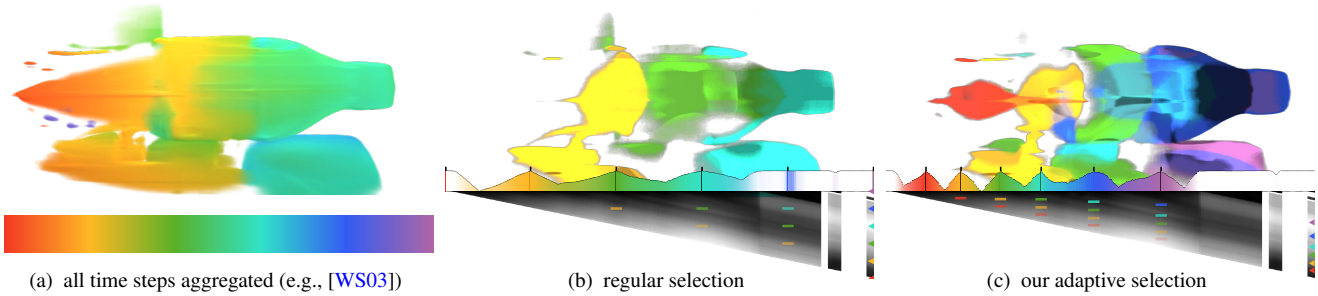
2. Related Work

Visualization of Time-Varying Volume Data. A survey of the visualization and visual analysis of multifaceted scientific data in general was given by Kehrer and Hauser [KH13]. Joshi and Rheingans [JR08] evaluate illustration-inspired techniques for time-varying data, like speedlines or flow ribbons. Bach et al. [BDA*16] review a range of temporal data visualization techniques, interpreting them as series of operations on a space-time cube. Likewise, Woodring and Shen [WWS03] consider time-varying volumetric data as a four-dimensional data field and apply high dimensional slicing and projection techniques. Woodring and Shen [WS03] further presented Chronovolumes, a static, direct rendering technique for time-varying data integrating all data over time in one volume. In contrast, our approach renders just a selection time steps to retain spatial shape information, and samples volumes individually to achieve higher quality via post-classification (cf. [HKRs*06], Sec. 6a). Jang et al. [JEG12] employ functional representations and efficient encoding to accelerate rendering. Balabian et al. [BVMG08] use temporal transfer functions and compositors (e.g., to highlight areas of high change). Lee and Shen [LS09a] visualize trend relationships among variables in multivariate time-varying data. Based on similarity matrices, Frey et al. [FSE12] detect and explore similarity in the temporal variation of field data.

While we directly work with scalar volume data, a large body of work in time-dependent volume visualization is based on feature extraction. Widanagamaachchi et al. [WCBP12] employ feature tracking graphs. Lee and Shen [LS09b] visualize time-varying features and their motion on the basis of time activity curves (TAC) that contain each voxel's time series. Fang et al. [FMHC07] use TACs in combination with different similarity measures. Silver et al. [SW97] isolate and track representations of regions of interest. The robustness of this approach has been improved by Ji and Shen [JS06] with a global optimization correspondence algorithm based on the Earth Mover's Distance. Scale-space based methods and topological techniques have also been used here (e.g., [WDC*07, NTN15]). Schneider et al. [SWC*08] compare scalar fields on the basis of the largest contours. Lu and Shen [LS08] propose interactive storyboards composed of volume renderings and descriptive geometric primitives. Post et al. [PVH*03] and McLoughlin et al. [MLP*10] give an overview for respective approaches in flow visualization. Here, among others, recent work focuses on glyphs [HLNW11], visual exploration [vPOBB*11], and efficient rendering [SZM*14].

In our approach, a transfer function may be applied to the volumes to define the (visual) importance of density values. While we manually generated the ones used in this paper, approaches have been proposed to (semi-)automatically generate transfer functions for time-varying data. They can be classified into data- or image-centric solutions [PLB*01] (i.e., parameters are derived from the volume or respective renderings). Jankun-Kelly and Ma [JKM01] give an overview on early work. Woodring and Shen [WS09] employ temporal clustering and sequencing to find dynamic features and create a corresponding transfer function. Akiba et al. [AFM06] use time histograms that are partitioned w.r.t. temporal coherency.

Comparison and Selection of Time Steps. Our flow-based distance metric is conceptually based on the earth mover's distance (EMD, also known as the Wasserstein metric). The EMD basically determines the minimum cost of turning one (mass) distribution into the other. It is particularly popular in the computer vision community where distances are typically computed between color histograms, e.g., for image retrieval [RTG00]. It works by constructing a complete graph (e.g., each pair of histogram bins is connected by an edge), and then using a minimum-cost flow solver to compute the distance (e.g., [GT90]). Bonneel et al. [BvdPPH11] decompose distributions into radial basis functions, and then apply partial transport that independently considers different frequency bands to interpolate between BRDFs, environment maps, stipple patterns, etc. Tong et al. [TLS12] use different metrics to compute the distance between data sets, and employ dynamic programming to select the most interesting time steps accordingly. In contrast to our approach, the aforementioned techniques use fully connected complete graphs for distance computation, which becomes prohibitively costly when considering a larger number of elements (cf. Sec. 7c for a comparison). Tong et al.'s selection approach is also excessively expensive in a number of cases (e.g., when choosing relatively few time steps from a long series). Therefore, we propose a novel selection procedure and employ progressive refinement. While we directly use volume data and its scalar values, Wang et al. [WYM08] extract a feature histogram per volume block (typically hundreds to thousands of voxels). While this allows them to consider higher-dimensional values, e.g., including gradient magnitude and direction, we can adequately handle small-scale movement, e.g., from a thin surface (cf. Fig. 8a). They derive entropy-based importance curves (similar to TACs) that characterize the local temporal behavior of each block. These are then classified via k-means, and the time range is partitioned into segments with equal accumulated importance values. One time step is chosen from each segment, taking into account the previously selected step (the first time step is always selected). Basically, this



(a) all time steps aggregated (e.g., [WS03]) (b) regular selection (c) our adaptive selection

Figure 2: Rendering the Bottle data set with (a) all time steps, and $|S| = 6$ time steps ((b) selected regularly and (c) with our approach).

relies on having a globally valid importance number for each time step. In contrast, we solely consider the mutual distances between all time steps (not just selected ones) to directly optimize the coverage of the whole time series (Wang et al. also discuss that their measure conceptually deviates from distances between time steps). The field of video analysis also deals with related analysis problems, yet typically employing different methodologies. Specialized image and video metrics are used to compare frames (e.g., [SB10]), and distinct approaches were proposed to generate summaries of videos, e.g., based on the motion of actors over time [CM10].

3. Overview and Structure

(a) Introductory Example. We now exemplify our approach with captured data from a laser pulse shooting through a bottle (Velten et al. [VWJ*13], cf. Sec. 7). In Fig. 2 (a), we map each time step to a different color and aggregate over time. While it roughly depicts the overall progress, the local shape of the pulse spreading out is lost. The colormap is also not fully utilized: it covers time ranges with basically nothing happening (particularly toward the end). In (b), we selected time steps with a regular spacing in time. While in the composited visualization individual time steps show the shape of the pulse, only a very sparse overview is given, and three out of the six selected time steps are practically empty. Through the adaptive selection with our approach (c), the overall progress can be seen in much more detail, showing the different stages that occur in the time series. The similarity chart (b and c, bottom) additionally depicts distances between time steps (similarity is indicated by color, where white represents the "empty" volume, cf. Sec. 6b).

(b) Main Components. Alg. 1 gives an overview of our approach. Initially, the set of considered time steps T^*, the set of selected time steps $S_{[]}$, and the map D storing the distances between time steps $t \in T^*$ are empty (Lines 2–4). In each iteration of our selection update loop, we obtain a new set of of time steps ΔT, either **(i)** as a subset $\Delta T \subset T - T^*$ from the full set of all time steps T (Line 6), or **(ii)** from an input stream delivering later time steps, e.g., from a concurrently running simulation (Line 7). In the remainder of the paper, we concentrate on **(i)**, the adaptive loading of time steps (Sec. 3c). However, we give an outlook on **(ii)** in Sec. 7f. Here, while we could also run just one refinement iteration (i.e., $\Delta T = T$), it reduces the time until (preliminary) selection results are computed. If ΔT is empty (i.e., $T^* = T$ already), we are done (Line 8). Otherwise, we integrate ΔT into T^* (Line 9), and update the distance map D with

Algorithm 1 Our approach for flow-based temporal selection.

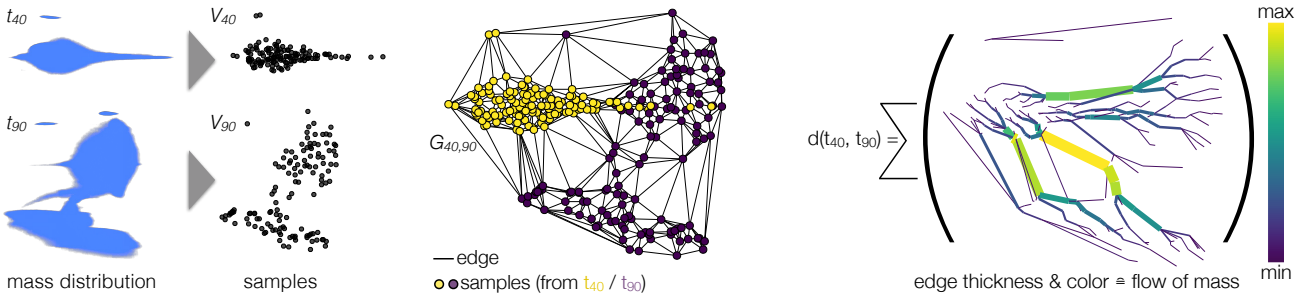
```
1:  procedure FLOW-BASED TEMPORAL SELECTION
2:    T* ← {}                                   ▷ set of considered time steps
3:    S[] ← {}                                  ▷ set of selected time steps
4:    D ← {:}                                   ▷ map of distances between time steps
5:    loop                                      ▷ progressively incorporate time steps (via (i) or (ii))
6:      (i) ΔT ← PROGRESSIVELOADING(D, T, T*)   ▷ ↓storage (Sec. 3c)
7:      (ii) ΔT ← INPUTSTREAMBUFFER(·)          ▷ ↓stream (Sec. 7f)
8:      if ΔT = {} then return                  ▷ exit if there are no new time steps
9:      T* ← T* ∪ ΔT                            ▷ merge with current set of time steps
10:     for all t_a ∈ ΔT do                     ▷ update distance map
11:       for all t_b ∈ T* do
12:         D ← D ∪ {(t_a, t_b) : FLOWDISTANCE(t_a, t_b)}   ▷ (Sec. 4)
13:     S[] ← SELECTION(D, S[])                 ▷ select time steps (Sec. 5, Alg. 2)
14:     S ← S_k ▷ visualization w/ k selected time steps (user-defined,Fig.5)
15:     VOLUMERAYCASTING(S)                     ▷ render volumes (Sec. 6a, Alg. 3)
16:     SIMILARITYCHART(D, S)                   ▷ show selection & distances (Sec. 6b)
```

the flow-based distances between all newly added time steps $t_a \in \Delta T$ and all currently considered time steps $t_b \in T^*$ (Lines 10-12, Sec. 4). Based on D and the previous selection set $S_{[]}$, we update $S_{[]}$ with the goal to optimally cover the full time series T (Line 13, Sec. 5). We visualize the selection consisting of $k = |S|$ selected time steps (Line 14), both directly with volume rendering (Line 15, Sec. 6a) and the respective distances via similarity charts (Line 16, Sec. 6b).

(c) Progressive Loading of Time Steps. We add new time steps $\Delta T \subset T$ to our current set $T^* \subset T$ with an adaptive selection scheme (PROGRESSIVELOADING(D, T, T^*), Line 6). Initially, we use the first and last time step in T, partition the range in between into equally-sized intervals, and randomly choose one time step from each. Later, the distances $d(t_a, t_b) \in D$ between subsequent considered time steps ($t_n, t_{n+1} \in T^*$) define a discrete probability distribution P that assigns a probability $p_t \in P$ to each $t \in T - T^*$:

$$p_t = \frac{d(\lfloor t \rfloor_{T^*}, \lceil t \rceil_{T^*})}{\lceil t \rceil_{T^*} - \lfloor t \rfloor_{T^*} - 1}. \tag{1}$$

Here, $\lfloor \cdot \rfloor_{T^*}$ and $\lceil \cdot \rceil_{T^*}$ denote the surrounding previous and next time step in T^*, respectively. Basically, a higher weight is applied between far-apart time steps $t_n, t_{n+1} \in T^*$, whereas each time step within such an interval has equal chances of getting picked. Then, we simply sample P randomly to obtain new time steps ΔT.



(a) **Sampling** of mass distributions (b) **Connection** of samples of two time steps (c) **Flow** gives mass movement along connections

Figure 3: Distance computation between two time steps at the example of time steps t_{40} and t_{90} of the Bottle data set.

4. Time Step Difference

We employ a flow-based distance metric that directly accounts for the (global) movement of mass. Mass is given by the scalar values of the volume data set (potentially after transfer function mapping). In contrast to the flow-based approach, an element-based distance metric only assesses differences for each element in space locally (e.g., individually for each grid cell). In other words, a flow-based method quantifies how much mass is moved how far, while an element-based method only assesses how much has changed at a certain location (disregarding the involved shift, cf. Sec. 7d). We efficiently compute flow-based distances between two time steps $t_a, t_b \in T^*$ in three basic steps (cf. Fig. 3, discussed in detail below):

(a) *Sampling* of volumes of t_a and t_b to yield samples V_a and V_b.
(b) *Connection* of V_a and V_b to obtain flow graph $G_{a,b}$.
(c) *Flow* computation solving graph $G_{a,b}$ to yield $d(t_a, t_b)$.

(a) Sampling ($t_a \to V_a, t_b \to V_b$, Fig. 3a). Each time step $t \in T$ is represented by a scalar volume $M_t : \mathbb{R}^3 \to \mathbb{R}$. To M_t, a user-defined transfer function ($\mathbb{R} \to \mathbb{R}$) may be applied to define value ranges of interest for the analysis (as well as their respective visual weight). In the following, we denote and handle M_t conceptually as a generic mass distribution. First, we sample M_t to yield a set of samples V_t with a fixed number of elements $|V|$. For this, we consider M_t as a discrete probability distribution, and sample it by (1) randomly generating numbers in the range $[0, \sum_{M_t}]$, and (2) choosing the associated elements in M_t via the prefix sum of M_t (this is done similarly for the adaptive loading of time steps in Sec. 3c). For the subsequent steps, only these samples V_t need to be stored (and M_t may be discarded), which is crucial for efficiently processing large volume data sets. Most importantly, a fixed number of samples $|V|$ across time steps directly delivers "balanced" problems for flow computation without further actions required (cf. (b, c)). It further yields easier-to-predict performance across different data sets, not only with respect to storage space but particularly regarding the required time for distance computation. While a large number of samples $|V|$ increases accuracy, it also increases storage cost as well as the computational cost of the subsequent *Connection* and *Flow* steps (cf. Sec. 7a for a closer discussion and evaluation). V_t needs to be regenerated when the transfer function is changed non-uniformly (plain scaling does not change the normalized mass distribution M_t).

(b) Connection ($(V_a, V_b) \to G_{a,b}$, Fig. 3b). To compute the distances between two time steps t_a and t_b, we now generate a flow graph $G_{a,b}$ on the basis of the sample sets V_a and V_b. $G_{a,b}$ contains edges $e \in E_{a,b}$ between these samples, where each edge is weighted $w(e)$ with the L1 distance between the respective sample pair (i.e., $G_{a,b}$ is weighted and undirected). We assign mass $m(v_a) = 1$ to all samples $v_a \in V_a$, while samples in $v_b \in V_b$ get $m(v_b) = -1$ (such that $\sum_{v \in V_a \cup V_b} = 0$ as $|V| = |V_a| = |V_b|$). For efficiency, we consolidate samples from the same spatial location ϕ, i.e., we replace them with a new vertex v_ϕ with $m(v_\phi) = \sum_{v \in V_a(\phi) \cup V_b(\phi)} m(v)$.

Typically, the EMD uses a bipartite graph that is generated by adding one edge for each pair of samples (v_a, v_b), with $v_a \in V_a$ and $v_b \in V_b$ (cf. Sec. 2). While this allows to determine an optimal solution (i.e., finding an assignment $V_a \to V_b$ with minimal associated cost), it also results in a total of $|V_a| \cdot |V_b|$ edges. This mainly causes the comparably high complexity of EMD, and makes it prohibitive to use for larger number of samples (e.g., there are millions of edges for thousands of samples already, cf. Fig. 7 and Sec. 7a). We therefore propose a different approach that significantly decreases the number of generated edges E, while still delivering close-to-optimal solutions. We add edges between the combined set of samples $V = V_a \cup V_b$, but only connect proximate vertices directly via 3D Delaunay triangulation (we employ CGAL's implementation [JPT15], Fig. 3b). This yields nice properties for our resulting graph with a relatively low number of edges $|E|$. First, the nearest neighbor graph is a subgraph of the Delaunay triangulation, which is important for close samples. Second, the shortest path between two vertices along Delaunay edges cannot be longer than $\frac{4\pi}{3\sqrt{3}} \approx 2.42$ times the Euclidean distance between them (e.g., [CDS12]).

(c) Flow ($G_{a,b} \to d(t_a, t_b)$, Fig. 3c). With our flow graph $G_{a,b}$, the remaining step is now to solve the respective minimum-cost flow problem: finding the cheapest possible way of turning V_a into V_b by transporting masses over the edges E. To achieve this, a solver computes a flow F that determines how much mass is transported over each edge $e \in E$ (in which direction). The solver does so in a way such that (1) applying F to V_a turns it into V_b, and that (2) the associated cost $d(t_a, t_b)$ is minimal, with

$$d(t_a, t_b) = \sum_{e \in E} F_{a \to b}(e) \cdot w(e). \tag{2}$$

To compute $F_{a \to b}$, we apply Google's implementation of a cost-scaling push-relabel algorithm to our graph $G_{a,b}$ [Goo15]. It maintains a preflow and gradually converts it into a maximum flow by moving flow locally between neighboring vertices using push operations considering an admissible network maintained by relabel operations [GT90]. We have no capacity constraints on the edges as in more general formulations of the minimum-cost flow problem.

Algorithm 2 Optimization of the set of selected time steps $S_{[]}$ (S_k gives best determined selection with k selected time steps, $\gamma(S_1) \leq \gamma(S_2) \leq \ldots \leq \gamma(S_{\sigma-1}) \leq \gamma(S_\sigma)$, cf. Fig. 5).

1: **function** SELECTION($D, S_{[]}$) ▷ best selection of $1 \ldots \sigma$ time steps
2: **for** $p \in \{1 \ldots n\}$ **do** ▷ run n iterations to improve $S_{[]}$
3: $S^* \leftarrow \{\}, \Gamma \leftarrow \{\}$ ▷ initialize selection set and distance buffer
4: **for** $k \in \{1 \ldots \sigma\}$ **do** ▷ incrementally add σ time steps
5: **for all** $t \in T^* - S^*$ **do** ▷ loop over time steps not in S^*
6: $\Gamma_t \leftarrow \gamma(S^* \cup t)$ ▷ compute distance for each t
7: **if** $\min_\gamma(\Gamma) < \gamma(S_s)$ **then** ▷ smallest distance in Γ is best overall?
8: $S_k \leftarrow S^* \cup \min_t(\Gamma)$ ▷ update best selection of s time steps
9: $\Gamma' \leftarrow 1 - \frac{\Gamma - \min_\gamma(\Gamma)}{\max_\gamma(\Gamma) - \min_\gamma(\Gamma)}$ ▷ normalize
10: $S^* \leftarrow S^* \cup \text{weightedRandomSampling}_t(\Gamma')$ ▷ randomly choose
return $S_{[]}$

5. Distance-based Time Step Selection

We aim to select time steps $S \subset T$ such that they optimally cover the whole time series T. For this, we quantify the coverage of each time step $t \in T$ via its minimum distance to neighboring selected time steps $s \in S$ (we also take special means to account for time steps that exhibit low total mass or that have not been loaded yet). Overall coverage is then computed as the sum of of all individual coverage values (a). On this basis, we generate not just one but a set of selections $S_{[]}$ consisting of $1 \ldots \sigma$ time steps for flexible interactive exploration. For this, we employ an iterative optimization approach: in each iteration, starting from $k = 1$ time steps, we incrementally add time steps to a selection candidate S^*, and update selection $S_k \leftarrow S_k^*$ if its coverage is better than with current best selection S_k (b). We propose an iterative Monte Carlo-based approach as computing the optimal solution typically causes excessive cost in practice (there are $\binom{|T|}{|S|}$ possibilities to select $|S|$ time steps from a time series of length $|T|$). For small problems, we compare our approach against the optimal solution in Sec. 7c.

(a) Time Series Coverage. The basis of our selection approach is a metric that quantifies the coverage of T with a subset $S \subset T$. For this, apart from $t \in S$, we also consider the empty volume ε with zero mass to adequately account for time steps of negligible total mass (i.e., that are (almost) fully transparent in terms of volume visualization). We do not want to explicitly have to store and visualize these, even if they exhibit a large distance to other time steps. Accordingly, the empty volume ε is never part of a selection S, but only considered in our criterion to quantify coverage. Using this, our coverage criterion $\gamma(S)$ is computed by the sum of the squared minimum distances of each time step $t \in T$ to its surrounding selected time steps ($\lfloor \cdot \rfloor_S$ & $\lceil \cdot \rceil_S$) and ε:

$$\gamma(S) = \sum_{t \in T} \min(d(\lfloor t \rfloor_S, t), d(t, \lceil t \rceil_S), d(t, \varepsilon))^2 / |T|. \quad (3)$$

$\lfloor \cdot \rfloor_S$ and $\lceil \cdot \rceil_S$ denote the next lower and upper time step in S. If there is no lower or upper neighbor, the distance computation yields $d(\cdot, t) = \infty$. For all $t \in T^*$ the required distances d have been computed previously and stored in map D (cf. Alg. 1, Lines 10–12). Additionally, we need distances to *(1)* the empty volume ε, and *(2)* the time steps $t \in T - T^*$ that have not been loaded yet.

(1) $d(t, \varepsilon)$. We compute the total mass $m_t = \sum_{m \in M_t} m$ of mass distribution M_t (cf. Sec. 4a), and define the distance to the empty volume to be $d(t, \varepsilon) = m_t \cdot \omega$ (i.e., the weighted total mass). Here, ω defines the ratio w.r.t. a maximally possible total mass of 1 (a fully opaque volume, in terms of volume visualization). Conceptually, it expresses how much mass is still regarded to be (visually) significant. We determined a value of $\omega = 0.0001$ to be generally useful via experiments and employ it throughout this paper.

(2) $d(t, t^*)$ for $t \in T - T^*, t^* \in T^*$. We first determine the surrounding considered time steps $\lfloor t \rfloor_{T^*}$ and $\lceil t \rceil_{T^*}$ ($\lfloor \cdot \rfloor_{T^*}$ and $\lceil \cdot \rceil_{T^*}$ denote the next lower and upper time step in the set of considered time steps T^*). Based on this, we then estimate the distance $d(t, t^*)$ as follows for $t \in T - T^*, t^* \in T^*$:

$$d(t, t^*) = \frac{d(\lfloor t \rfloor_{T^*}, \lceil t \rceil_{T^*})}{\lceil t \rceil_{T^*} - \lfloor t \rfloor_{T^*}} \cdot \begin{cases} (\lceil t \rceil_{T^*} - t) & \text{if } t^* < t \\ (t - \lfloor t \rfloor_{T^*}) & \text{else} \end{cases} + \begin{cases} d(\lfloor t \rfloor_{T^*}, t^*) & \text{if } t^* < t \\ d(\lceil t \rceil_{T^*}, t^*) & \text{else} \end{cases}. \quad (4)$$

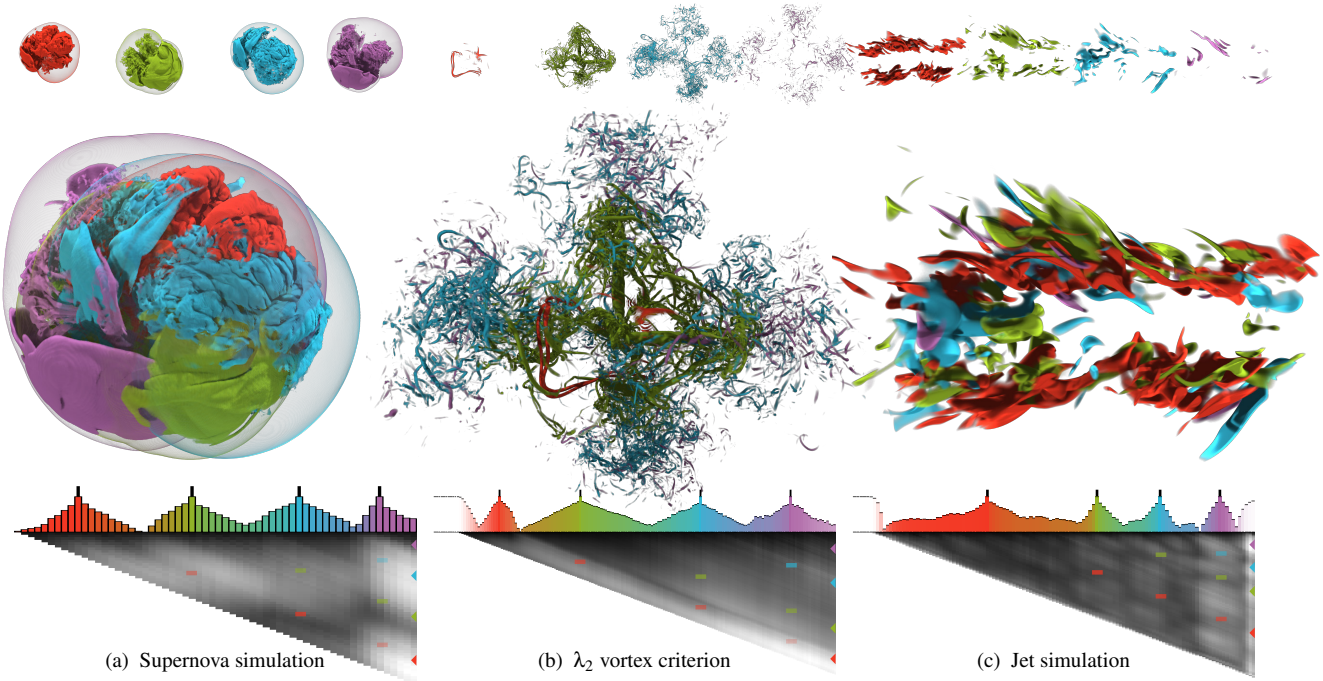
This means that to compute $d(t, t^*)$ with $t^* > t$, we first estimate the distance of t to $\lceil t \rceil_{T^*}$ via linear interpolation (Eq. 4 *top*), and then add the distance of $\lceil t \rceil_{T^*}$ to t^* (Eq. 4 *bottom*) (vice versa for $t^* < t$). For each required distance computation at least one time step is an element of T^*, as no timestep $t \notin T^*$ can be part of selection S.

(b) Selection Approach. Conceptually, we aim to minimize the distance of the selected time steps S to the full set of time steps T:

$$\min_{S \subset T^*} (\gamma(S)). \quad (5)$$

Our approach toward this goal generates not just one but a set of selections $S_{[]}$ consisting of $1 \ldots \sigma$ time steps (σ being a user-defined value). Providing this range of selections $S_{[]}$ together with their respective distance values γ allows to choose one selection S_k ($k \in \{1 \ldots \sigma\}$) from this range a posteriori (Alg. 1, Line 14). Basically, we execute several iterations of our optimization scheme (Alg. 2, Line 2), each of which generates selections S^* consisting of $1 \ldots \sigma$ time steps. In this process, we initially assign the empty set to S^* (Line 3), and then iteratively add time steps to it in the following (Line 4). For each number of selected time steps k, we evaluate the distances of $S^* \cup t$ for all time steps $t \in T^*$ that are not yet in S^* (Line 5). The result for each time step t is then stored in a map Γ (Line 6). For the sake of efficiency, we only evaluate the changed parts of S^* w.r.t. the previous iteration. We achieve this by caching the sum of distances $d_\Sigma(s_a, s_b)$ between pairs of subsequent time steps $s_a, s_b \in S^* \cup \{t_0, t_{|T|-1}\}$: $d_\Sigma(s_a, s_b) = \sum_{t \in \{s_a, \ldots, s_b\}} \gamma_t(S)$.

The result map Γ is then used for two purposes. First, we select the time step t that yields the smallest value, and potentially update $S_{[]}$ accordingly (Lines 7 & 8). Second, we add a time step to our current set S^* that is then used as basis for the next iteration. For this, we first transform the contents of our distance map Γ to Γ' such that the largest distance translates to zero and the smallest distance to one (Line 9). We then use weighted random sampling on Γ' to determine the time step that is added to S^* (Line 10). In essence, this means that theoretically all time steps (except for the one with the worst coverage) could be chosen, yet the respective likelihood increases with their impact on the coverage of the time series.



(a) Supernova simulation (b) λ_2 vortex criterion (c) Jet simulation

Figure 4: Selections for $|S| = 4$. The selected time steps of the Supernova data set (a) are rather evenly distributed, representing the underlying steady development. Both in the λ_2 (b) and the Jet data set (c), values of interest do not emerge immediately from the start of the series. While in (b) vortices develop in the center and evenly disaggregate outward, (c) the Jet starts slow, but rapidly changes its structure toward the end.

6. Visualization

We present individual and integrated volume renderings of selected time steps S (cf. Fig. 4 *top* & *center*) and give the distances between time steps via our similarity chart (cf. Fig. 4 *bottom*). A user may also interactively select time steps S. The similarity chart both directly visualizes the distance map D and depicts the coverage $\gamma_t(S)$ provided by the selected time steps S for each time step t.

(a) Individual and Integrated Volume Rendering. We provide both renderings $\chi^{0\ldots|S|-1}$ of all selected time steps S individually as well as an integrated volume visualization χ of all time steps in S (Alg. 3). A 3D atlas texture stores the volumes of time steps $\in S$. We employ front-to-back volume raycasting (Line 3). For each sampling position p, we loop over all selected volumes S (Line 5). We fetch the color rgb and opacity α for the respective sample via $m_{\text{rgb}\alpha}$ (Line 6). The mapping of color depends on the order i of the respective selected time step $s_i \in S$ (we employ a color progression from red over green and blue to violet, Fig. 2a *bottom*). The opacity α is directly taken from the mass distribution of a time step. With rgb and α, we update the entry χ^i of the individual renderings (Line 7). We employ shadow and lighting operations (sl) to improve the visual comprehensibility of structures. We further use a shadow map that was pre-computed by sending shadow rays to each light source (it is updated as soon as the mass distribution or the selection changes). It stores a separate shadow entry for each selected volume $t \in S$ and each light source using the same atlas structure employed for the volume data. We use the individual entries for each selected time step to generate the individual time step renderings (Line 7, Fig. 4 *top*), and later the combination of all shadow rays to yield the integrated rendering (Line 11, Fig. 4 *center*).

For the combined rendering, the blending order of integration through time can have a significant visual impact when different regions of interest in time are co-located in space. The occurring effect of earlier time steps occluding older time steps has been denoted as temporal occlusion [WS03] (cf. Fig. 9b). Woodring and Shen [WS03] work around this issue by letting the user explicitly assign opacities to different time steps to specify their importance. Instead, we employ a step slicing approach to adequately composit the values from different volumes: we scale the opacity contribution of each time step with a (conceptually infinitesimally) small Δ (Line 8). Here, we limit the maximum opacity to a value that is slightly below 1 to achieve an adequate combination with the other time steps even if some of them are fully opaque. We then convert from premultiplied to straight color (Line 9), stretch the opacity contribution of the step back again from Δ to 1 (Line 10), and composit the result (Line 11). We terminate our ray early if all individual entries χ exhibit a large enough opacity value (Line 12).

(b) Similarity Chart. Similarity charts (e.g., Fig. 4 *bottom*) consist of two components: the selection chart and the distance plot. *Selection charts* (top half) depict selected time steps S and show the coverage of the time steps in between. Each time step is represented by a bar, whose height represents the smallest distance $d(\cdot,\cdot)$ of the respective time step to either of its two neighbors or the empty volume ε (cf. Eq. 3). With the maximum distance $d_{\max}$ occurring in a time series, we determine the height h of a bar as follows: $h = \hat{h}(1 - d(\cdot,\cdot)/d_{\max})$ (i.e., $\gamma_t(S) = 0$ leads to a selection bar of maximum height $\hat{h}$ for time step t, while the maximum distance $d_{\max}$ yields a zero-height bar). The color of the bars represents which selected time step (or the empty volume ε) the respective time step has the smallest distance to. For selected time steps $t \in S$, this

Algorithm 3 Volume rendering of selected time steps S for each step along a ray during raycasting (w/ ray step size 1). We use slicing step size $\Delta << 1$. $m^i_{\text{rgb}\alpha}(\cdot)$ gives color w.r.t. the selection index i, and opacity α is obtained from the mass at the current position p. $\text{sl}_i(\cdot)$ gives the results of shadowing and lighting for $s_i \in S$.

```
1:  procedure VolumeRaycasting(S)
2:    χ, χ^0, ..., χ^{|S|-1} ← (0,0,0,0)
3:    for all p ∈ P_ray do                          ▷ sample front-to-back along ray
4:      χ' ← (0,0,0,0)                              ▷ integrated sample for combined rendering
5:      for i ∈ 0...|S|-1 do                        ▷ integrate over all selected time steps
6:        rgb, α ← m^i_{rgbα}(p)                    ▷ lookup color and opacity for i
7:        χ^i ← over(χ^i, (sl_i(rgb, p), α))        ▷ composit for individual render
8:        χ' ← over(χ', (rgb, 1 − (1 − α)^Δ))       ▷ composit (step size Δ)
9:      χ'_rgb ← χ'_rgb / χ_α                       ▷ convert from premultiplied to straight color
10:     χ'_α ← 1 − ∛(1 − χ'_α)                      ▷ stretch contribution from Δ to 1
11:     χ_Σ ← over(χ, (sl(χ_rgb, p), χ_α))          ▷ composit combined rendering
12:     if min(χ_α^{0...|S|-1}) > 0.98 then         ▷ early ray termination
13:       break
```

color comes directly from m_{rgb} (cf. (a)). For $t \in T^* - S$, we mix the colors weighted by their distance, with the empty volume ε being represented by white. Selected time steps $t \in S$ are indicated via an extra black line through their middle. For each time step, the *distance plot* (bottom half) depicts the distances to all previous time steps (in the case of progressively loading time steps, missing distances are interpolated for display). High and low opacity (of black pixels) denote high and low similarity, respectively. We also reduce the opacity in case of low distance to the empty volume ε. Distance plots can help to identify recurrent patterns in the data (cf. Fig 1). In the plot, the distance between selected time steps is highlighted via colored boxes (w/ low opacity $\hat{=}$ large distance). For orientation, fully opaque arrows also indicate selected time steps along the y-axis to the right. Elements in the distance plot may be selected directly during interactive analysis—using the left and right mouse button to pick a time step w.r.t. the x- and y-axis, respectively—to analyze temporal patterns (cf. Fig. 8).

7. Results

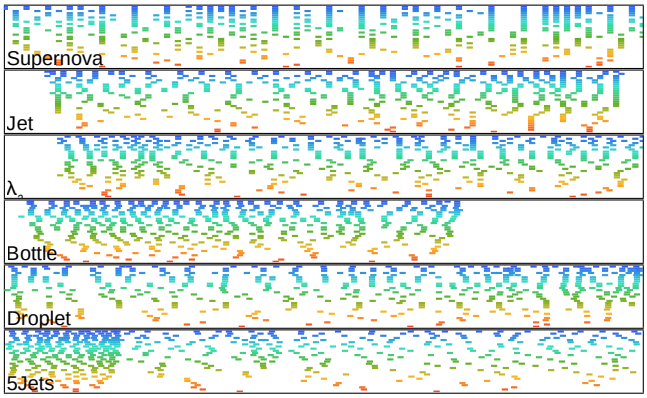
We use six data sets in our evaluation. The 5Jets data set (128^3, 2000 time steps) results from a simulation of five jets entering a rectangular region (Fig. 1). The Bottle data set (resolution 900×430, 465 time steps) depicts a laser pulse shooting through a bottle, captured via Femto Photography (Velten et al. [VWJ*13], Fig. 2). The Supernova data set (432^3, 60 time steps) results from a supernova simulation (Fig. 4a). Next, the λ_2 data set (529^3, 174 time steps) contains the λ_2 vortex extraction criterion applied to a CFD simulation (Fig. 4b). The Jet data set ($480 \times 720 \times 120$, 121 time steps) stems from a turbulent combustion simulation (Fig. 4c). 5Jets, Supernova and Jet were obtained from http://vis.cs.ucdavis.edu. The Droplet data set (256^3, 1000 time steps) contains simulation results of two drops colliding asymmetrically (Fig. 10, courtesy of C. Meister, Institute of Aerospace Thermodynamics, University of Stuttgart). For it, we use the data values directly as mass, while a custom transfer function is used for the others. We supply full time series renderings of each data set as well as an interaction demo in our accompanying video. We used an NVIDIA GTX980, an Intel Core i7-4770, and 16GB of memory in our evaluation. While volume visualization uses the GPU via CUDA, distance computation and time step selection run in parallel on the CPU with eight threads using OpenMP. Throughout our evaluation, we used several settings that—according to our experiments—generate good results at acceptable run times: we took $|V| = 4096$ samples from each time step for distance computation, weighted the empty volume ε by $\omega = 0.0001$, and rendered at a resolution of 1920×1200. For time step selection, we used $n = 131072$ runs for one execution of Selection$(\cdot)$ in the non-progressive case. In the progressive case, Selection$(\cdot)$ executes every time new time steps are loaded. As it does not start from scratch but iteratively improves selection $S_{[]}$, we used only $n = 4096$ runs per execution. These settings are chosen conservatively, i.e., after the specified number of iterations, the selection typically did not change anymore in our experiments (and if, only negligible improvements were achieved). In general, volume render times vary with the resolution of the data set, the number of selected time steps, as well as its mass distribution (due to early ray termination). It remains interactive for all considered data sets (and moderate $|S|$), e.g., rendering the Jet with $|S| = 6$ time steps takes ≈ 75 ms per frame, while the Supernova takes around 185 ms due to its larger resolution. In the following, we

data set	Supernova	Jet	λ_2	Bottle	Droplet	5Jets
#t time steps $\lvert T\rvert$	60	121	174	465	1000	2000
# distances $\lvert D\rvert$	1830	7381	15225	108345	500500	2001000
distance timings ($\sum$)	535 s	2454 s	4832 s	19621 s	164014 s	369419 s
distance timings ($\frac{\sum}{\lvert D\rvert}$)	0.29 s	0.33 s	0.31 s	0.18 s	0.32 s	0.18 s
selection timings ($\sum$)	98.5 s	214.9 s	317.42 s	943.6 s	2300.1 s	6526.5 s
selection timings ($\frac{\sum}{n}$)	0.0007 s	0.001 s	0.002 s	0.007 s	0.017 s	0.049 s

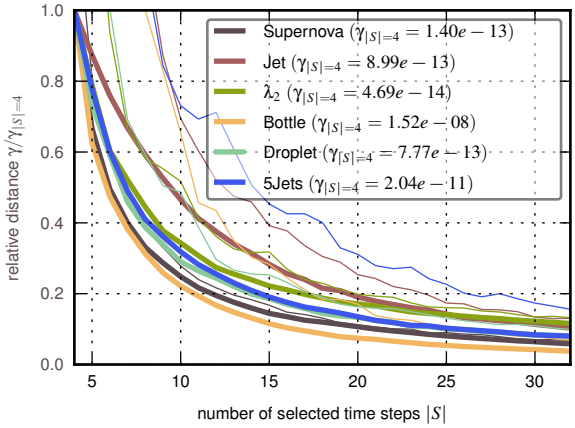
Table 1: Full distance and selection timings ($T^* = T, \sigma = 32$).

(a) demonstrate results of our flow-based distances and selection,
(b) discuss timings and the progressive incorporation of time steps,
(c) compare against optimal solutions w.r.t. distances and selection,
(d) compare against results achieved with element-based distances,
(e) compare our compositing scheme against alternatives,
(f) and finally discuss limitations and potentials for extensibility.

(a) Flow-based Distances and Selection (Figs. 4 & 5). We select time steps such that they adequately cover the whole time series. If the rate of change is relatively steady, then this results in a comparably uniform selection, as in the example with the Supernova data set (Fig. 4a). If, in contrast, the rate of change varies significantly, our selection process adapts to this by picking more time steps in intervals with a higher rate of change. This has been shown earlier with the 5Jets data set (Fig. 1): initially, five jets induce a wave, with a similar movement occurring later at a slower pace. Accordingly, more time steps are selected in the very beginning, with the later time range being represented much more sparsely. For λ_2, particularly high vorticity values are of interest: they only occur in a limited range of time steps, but evenly disaggregate outward within this time frame (Fig. 4b). In contrast, the Jet data set slowly develops a certain structure in the beginning, with rapid changes happening toward the end (Fig. 4c). Both in the λ_2 and the Jet data set early time steps are covered via empty volumes in temporal ranges exhibiting little to no mass (i.e., very low visual impact). This has also been demonstrated earlier with the Bottle data set where one laser pulse is traced, and basically nothing happens before and after (Fig. 2).



(a) Selection refinement from $|S| = 4$ (red/bottom) to $|S| = 32$ (blue/top).

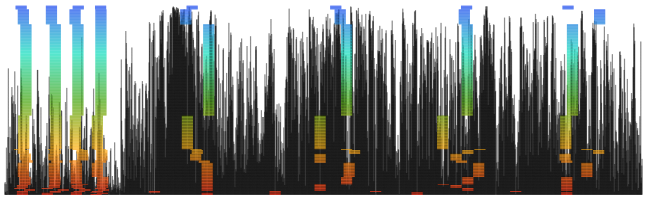


(b) Time series coverage $\gamma(\cdot)$ w.r.t. the number of selected time step in our selection (thick lines) and the uniform selection R (thin lines).

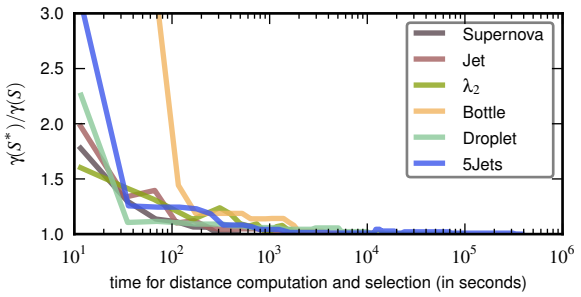
Figure 5: Selections $S_{[]}$ for our data sets from $|S| = 4$ to $|S| = 32$.

Our selection procedure generates a set of selections $S_{[]}$ consisting of $|S| \in \{1 \dots \sigma\}$ time steps. Each of these selections $S \in S_{[]}$ reflects the temporal development of the data, with a higher rate of refinement for increasing $|S|$ (Fig. 5a). Accordingly, our error evaluation criterion $\gamma(S)$ decreases with higher values of $|S|$. Overall, this happens at a similar rate across all considered data sets (Fig. 5b). While the uniform selection R also benefits from more selected time steps, the coverage is significantly worse in comparison to S.

(b) Timings & Progressive Refinement (Tab. 1, Fig. 6). To reach final selection results (i.e., for $T^* = T$), we require all mutual distances between considered time steps (and the empty volume ε): $t_0, t_1 \in T \cup \varepsilon$ and $t_0 < t_1$ (distance computations are commutative). Tab. 1 lists distance computation and selection timings for different data sets. The average performance of computing a single time step varies depends on the structure of the constructed flow graph. *Sampling* (that only needs to be done once per time step) and *Connection* only have negligible performance impact, with the major time share being consumed by the solver to determine the *Flow* (Sec. 4). Naturally, the total time for full distance computation and selection heavily depends on the number of time step $|T|$. To address the high full computation time for data sets with a high temporal resolution, our approach supports the progressive selection with continuously



(a) Selection refinement for 5Jets (black denotes missing time steps).



(b) Time spent for progressive refinement and achieved distance in relation to final result considering all time steps (i.e., for $T = T^*$).

Figure 6: Progressively adding time steps from T to T^* ($|S| = 8$).

data set	Supernova	Jet	λ_2	Bottle	Droplet	5Jets
(time steps)	60	60	87	60	66	66
distance timings (del.≙Delaunay triangulation; com.≙complete graph)						
time del.	518.4 s	599.7 s	1231.5 s	509.8 s	788.3 s	409.8 s
time com.	51 482.4 s	44 912.2 s	87 841.5 s	37 301.6 s	32 345.8 s	21 630.7 s
evaluation of distances computed with del. w/ com. as reference						
Δd avg	1.03	1.03	1.03	1.01	1.01	1.02
max d	1.08	1.10	1.10	1.08	1.05	1.10
selection timings with stochastic (sto.) and full optimization						
time sto.	10.7 s	10.5 s	16.21 s	10.6 s	14.6 s	11.8 s
time full	41.0 s	41.0 s	452.3 s	41.0 s	262.4 s	76.0 s
evaluation value γ of selection ($|S| = 6$; relative to com. full)						
del. & sto.	1.000	1.000	1.000	1.000	1.000	1.006
del. & full	1.000	1.000	1.000	1.000	1.000	1.000
com. & sto.	1.000	1.000	1.008	1.000	1.000	1.000
com. & full	1	1	1	1	1	1
regular	1.225	2.112	1.570	2.058	1.437	3.001

Table 2: Comparison of our approach against optimal solutions both for computing distances and selections (downsampled time range).

added time steps $T^* \subset T$. Fig. 6a depicts the continuous refinement of the selection over time as well as the incremental addition of new time steps. Fig. 6b shows the coverage (in relation to the final result) w.r.t. the time spent for selection and distance computation. A close approximation to the final result is achieved quickly after tens to maximally 250 seconds across all considered data sets.

(c) Comparison Against References (Tab. 2, Fig. 7). Next, we compare our approaches for distance computation and selection against respective optimal solutions (Tab. 2). For the distance computation, we construct the full bipartite graph that spans connections between all pairs of samples from different time steps. For the selection, we try all possible time step selection combinations (this also uses our optimization of only re-evaluating changed parts of the selection). As these approaches are significantly more costly than

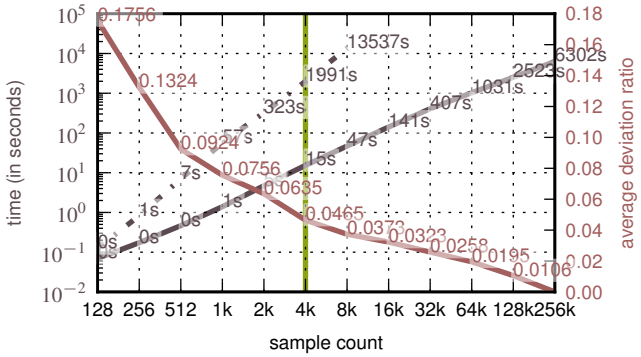
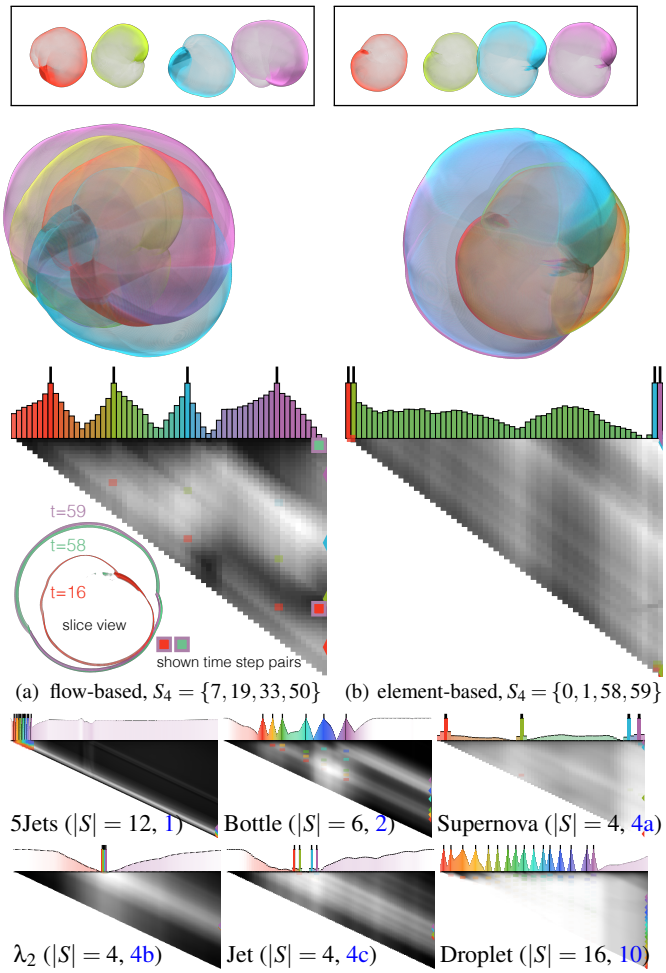


Figure 7: Timings (dark gray, solid: Delaunay triangulation, dotted: full graph) and accuracy (red, w.r.t. 256k results) for different numbers of samples with the Supernova data set. The green line depicts the sample count we use throughout this paper.

ours, we uniformly reduce the number of considered time steps. For the distance computation, our Delaunay graph-based approach is much faster in comparison to the reference complete bipartite graph (Tab. 2, *time del.* vs. *time com.*), while at the same time both average and maximum deviation are low (Tab. 2, Δd and $\max d$). Regarding the selection, even when only choosing $|S| = 6$ time steps of a set of 60, this already amounts to $\binom{60}{6} \approx 50M$ possible configurations. This leads to timing differences of several orders of magnitude of the full solution against our approximation (Tab. 2, *time sto.* vs. *time full*). Still, the achieved evaluation value $\gamma(S)$ of the approximation is very close to the reference solution (Tab. 2, *com.& sto.*). Also, using Delaunay flow graphs leads to basically the same selection results as complete flow graphs (Tab. 2, *del. & full*). The combination of both our approaches results in comparable quality w.r.t. the optimal solution, yet requires significantly less time (Tab. 2, *del. & sto.*).

Next, we evaluate the scaling of distance computations with different numbers of samples $|V|$. Fig. 7 shows both the deviation of different sample counts to a reference with 256k samples (red line) as well as the respective timings (dark gray) on a logarithmic scale. Our Delaunay-based approach performs significantly faster in comparison to the full graph (solid versus dotted line), with the gap widening with an increasing sample count (i.e., 4k samples with a complete graph takes approximately as long as 256k samples with our Delaunay-based graph). Also, the deviation to the reference continuously decreases with more samples. More than 80% of the total distance compute time is spent by the minimum-cost flow solver, while the remainder is almost exclusively due to Delaunay meshing.

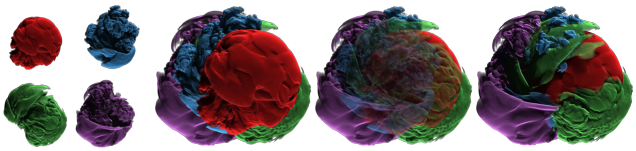
(d) Comparison against Element-based Distance (Fig. 8). To compare the outcome of the flow- and the element-based distance metric, we adjust the transfer function to only show the "hull" of the Supernova data set (Fig. 8). The distance plots show that the flow-based metric leads to gradual changes when comparing a time step against its predecessors, while for the element-based metrics this transition is very sharp (e.g., time steps 58 and 59 have one of the highest assigned mutual distances due to their high individual mass, yet they are very close in shape and position). The reason for that is that for the element-based distance even small shifts from one time step to the other yield basically the same results as two completely unrelated time steps if they only exhibit little overlap. In contrast,



(c) element-based selection w/ same $|S|$ as in flow-based examples

Figure 8: (a) Flow- and (b) element-based distances for the "hull" of the Supernova data set. In (a, *bottom left*), a slice view for three selected time steps is shown to investigate a detected similar process. (c) Element-based selection results for all data sets.

our flow-based metric detects (a series of) structurally similar time steps much more robustly. This is exemplified in Fig. 8a (*bottom left*): time steps 58 and 59 are very similar, and while the flow-based metric recognizes this, the element-based metric finds them to be completely different. From the distance plot of the flow-based metric (a), we further detect a similar process running approximately 40 time steps apart (dark line in the distance plot). Looking at time steps 16 and 58/59 indicates that this is due to a similarly shaped and oriented hull. Differences in the computed distances also show in the selection: while the flow-based distances lead to a fairly regular selection, reflecting the fairly even movement (see accompanying video), the element-based distances are not particularly expressive, leading to the selection of two time steps in the very beginning and end. This is due to the fact that these exhibit a relatively high total mass, which results in higher element-based distances overall when there is little overlap between the time steps. These issues can be seen to a different extent across all data sets (Fig. 8c).



(a) single (img) (b) over (img) (c) our (img) (d) our (vol)

Figure 9: Different approaches to visualize the selected timesteps: (a) individually, (b) blending images with the over-operator and (c) our compositing approach, and (d) ray-space compositing.

(e) Temporal Compositing (Fig. 9). We now compare different compositing approaches. Side-by-side renderings avoid temporal occlusion from other time steps [WS03]), yet relative positioning is hard to assess (Fig. 9a). Blending the renderings with the over-operator in image space also cannot achieve a correct impression of involved depth relations (b). Our compositing approach on images shares the same weakness, yet yields a clearer view on the details of each involved data set (c). Our ray-space compositing approach combines volumes during raycasting, and allows for correct depth assessment, supported by shading and shadowing (d).

(f) Discussion of Limitations and Extensibility. As we only consider surrounding selected time steps for coverage, our selection currently does not explicitly account for recurrent processes. While the user can easily detect such instances visually via our distance plot, reflecting this in the selection remains for future work. Also, while the integrated visualization allows to directly compare the spatial extent of different time steps, it generally does not scale to more than a couple of composited time steps due to temporal occlusion. However, this heavily depends on the data set: for λ_2 which is structurally complex for each individual time step already (Fig. 4b), a larger number of time steps both results in significant occlusion and visual clutter. In contrast, the Droplet data set due to its clear and sparse structure can take a much higher number of time steps (cf. Fig. 10). While a comparably high number of selected time steps is useful for the analysis of the 5Jets data set as well, temporal occlusion occurs for processes happening in the same spatial area at different points in time (Fig. 1). However, our approach allows to circumvent this in several ways: (1) similarity charts already give indications of such occurrences, (2) renderings of individual time steps are additionally shown, and (3) selections are computed with $|S| \in \{1 \dots \sigma\}$ time steps such that the number of shown time step $|S|$ can be interactively adjusted. While picking in the similarity charts to select time steps for visualization is supported already, more advanced interaction possibilities could further be added for improved visual analysis. Furthermore, although our Delaunay-triangulated graph design proved to be very efficient for distance computation, it is still the major cost factor in our approach due to the relatively high cost of the flow solver. However, as demonstrated above, our progressive approach mitigates by delivering good approximations of the final result early with few computed distances. Our approach could also be used in in situ and streaming application scenarios (cf. Sec. 3b). Conceptually, it works directly by using an input stream as drop-in replacement for adaptive data loading (**(ii)** instead of **(i)** in Alg. 1). However, changes are required when the total size of the incoming data becomes prohibitive to store in full (i.e., $|T^*| \leq c$ for some capacity limit c). To address this, an adjusted version of our progressive loading scheme (Sec. 3c) could be used that adaptively

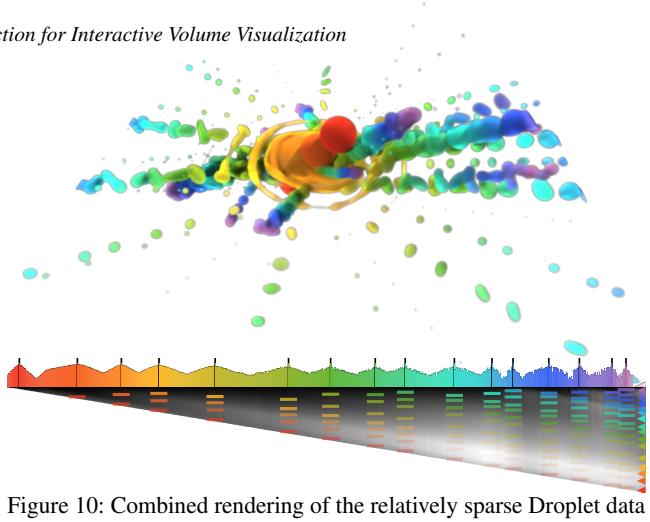


Figure 10: Combined rendering of the relatively sparse Droplet data set with comparably many selected time steps ($|S| = 16$).

updates our working set T^* whenever a new batch of time steps ΔT comes in: $T^* \leftarrow \text{AdaptiveStorage}(c, T^* \cup \Delta T, D)$.

8. Conclusion

We presented our approach to adaptively select and visualize time steps from time-dependent volume data for an integrated and comprehensive visualization. We directly operate on the volume representation, and require no domain or problem-specific knowledge (like for feature extraction). Our reduced set of selected time steps not only saves cost with respect to render time and storage, but also allows to show both spatial structure and temporal development in one rendering. The selection is employs a flow-based technique to determine meaningful distances between time steps. Both for distance computation and selection we present accelerated approaches that are significantly faster yet yield almost identical results w.r.t. reference solutions. We also compute selection sets consisting of different numbers of time steps alongside to support interactive exploration. Our approach further supports the adaptive progressive incorporation of new time steps, which achieves meaningful preliminary results very quickly (that are then further refined in the following). We show both individual and integrated renderings of the selected time steps, and visualize selections and computed distances via our new similarity chart.

For future work, we aim to apply our approach to in-situ visualization scenarios as outlined above. We want to further experiment with other (heuristic) solvers that might deliver higher performance with our flow graph setup. Flexibly adapting the number of samples taken from the data for distance computation could also lead to efficiency improvements. Additionally, we plan to account for detected recurrent processes in our selection, and extend interaction possibilities for improved explorative visual analysis.

Acknowledgments

The authors would like to thank the German Research Foundation (DFG) for supporting the project within project A02 of SFB/Transregio 161 and the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart.

References

[AFM06] AKIBA H., FOUT N., MA K.-L.: Simultaneous classification of time-varying volume data based on the time histogram. EUROVIS'06, Eurographics Association, pp. 171–178. 2

[BDA*16] BACH B., DRAGICEVIC P., ARCHAMBAULT D., HURTER C., CARPENDALE S.: A descriptive framework for temporal data visualizations based on generalized space-time cubes. *Computer Graphics Forum* (2016), n/a–n/a. 2

[BvdPPH11] BONNEEL N., VAN DE PANNE M., PARIS S., HEIDRICH W.: Displacement interpolation using lagrangian mass transport. *ACM Trans. Graph. 30*, 6 (2011), 158:1–158:12. 2

[BVMG08] BALABANIAN J.-P., VIOLA I., MÖLLER T., GRÖLLER E.: Temporal styles for time-varying volume data. In *Proceedings of 3DPVT'08 - the Fourth International Symposium on 3D Data Processing, Visualization and Transmission* (June 2008), Gumhold S., Kosecka J., Staadt O., (Eds.), pp. 81–89. 2

[CDS12] CHENG S.-W., DEY T. K., SHEWCHUK J.: *Delaunay Mesh Generation*, 1st ed. Chapman & Hall/CRC, 2012. 4

[CM10] CORREA C. D., MA K.-L.: Dynamic video narratives. *ACM Trans. Graph. 29*, 4 (2010), 88:1–88:9. 3

[FMHC07] FANG Z., MÖLLER T., HAMARNEH G., CELLER A.: Visualization and exploration of time-varying medical image data sets. In *Proceedings of Graphics Interface 2007* (New York, NY, USA, 2007), GI '07, ACM, pp. 281–288. 2

[FSE12] FREY S., SADLO F., ERTL T.: Visualization of temporal similarity in field data. *IEEE TVCG 18* (2012), 2023–2032. 2

[Goo15] GOOGLE: Google Optimization Tools, 2015. 4

[GT90] GOLDBERG A. V., TARJAN R. E.: Finding minimum-cost circulations by successive approximation. *Math. Oper. Res. 15*, 3 (1990), 430–466. 2, 4

[HKRs*06] HADWIGER M., KNISS J. M., REZK-SALAMA C., WEISKOPF D., ENGEL K.: *Real-time Volume Graphics*. A. K. Peters, Ltd., Natick, MA, USA, 2006. 2

[HLNW11] HLAWATSCH M., LEUBE P., NOWAK W., WEISKOPF D.: Flow radar glyphs & static visualization of unsteady flow with uncertainty. *IEEE TVCG 17*, 12 (2011), 1949–1958. 2

[JEG12] JANG Y., EBERT D., GAITHER K.: Time-varying data visualization using functional representations. *IEEE TVCG 18*, 3 (2012), 421–433. 2

[JKM01] JANKUN-KELLY T. J., MA K.-L.: A study of transfer function generation for time-varying volume data. In *Eurographics Conference on Volume Graphics* (2001), VG'01, pp. 51–66. 2

[JPT15] JAMIN C., PION S., TEILLAUD M.: 3D triangulations. In *CGAL User and Reference Manual*, 4.6.2 ed. CGAL Editorial Board, 2015. 4

[JR05] JOSHI A., RHEINGANS P.: Illustration-inspired techniques for visualizing time-varying data. In *Visualization, 2005. VIS 05. IEEE* (2005), pp. 679–686. 2

[JR08] JOSHI A., RHEINGANS P.: Evaluation of illustration-inspired techniques for time-varying data visualization. *Computer Graphics Forum 27*, 3 (2008), 999–1006. 2

[JS06] JI G., SHEN H.-W.: Feature tracking using earth mover's distance and global optimization. *Pacific Graphics* (2006). 2

[KH13] KEHRER J., HAUSER H.: Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE TVCG 19*, 3 (2013), 495–513. 2

[LS08] LU A., SHEN H.-W.: Interactive storyboard for overall time-varying data visualization. In *Visualization Symposium, 2008. PacificVIS '08. IEEE Pacific* (2008), pp. 143–150. 2

[LS09a] LEE T.-Y., SHEN H.-W.: Visualization and exploration of temporal trend relationships in multivariate time-varying data. *IEEE TVCG 15*, 6 (2009), 1359–1366. 2

[LS09b] LEE T.-Y., SHEN H.-W.: Visualizing time-varying features with tac-based distance fields. In *Visualization Symposium, 2009. PacificVis '09. IEEE Pacific* (2009), pp. 1–8. 2

[MLP*10] MCLOUGHLIN T., LARAMEE R. S., PEIKERT R., POST F. H., CHEN M.: Over two decades of integration-based, geometric flow visualization. *Computer Graphics Forum 29*, 6 (2010), 1807–1829. 2

[NTN15] NARAYANAN V., THOMAS D. M., NATARAJAN V.: Distance between extremum graphs. In *IEEE Pacific Visualization Symposium* (2015), pp. 263–270. 2

[PLB*01] PFISTER H., LORENSEN B., BAJAJ C., KINDLMANN G., SCHROEDER W., AVILA L. S., MARTIN K., MACHIRAJU R., LEE J.: The transfer function bake-off. *IEEE Comput. Graph. Appl. 21*, 3 (2001), 16–22. 2

[PVH*03] POST F. H., VROLIJK B., HAUSER H., LARAMEE R. S., DOLEISCH H.: The state of the art in flow visualisation: Feature extraction and tracking. *Computer Graphics Forum 22*, 4 (2003), 775–792. 2

[RTG00] RUBNER Y., TOMASI C., GUIBAS L.: The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision 40*, 2 (2000), 99–121. 2

[SB10] SESHADRINATHAN K., BOVIK A. C.: Motion tuned spatio-temporal quality assessment of natural videos. *IEEE Transactions on Image Processing 19*, 2 (2010), 335–350. 3

[SW97] SILVER D., WANG X.: Tracking and visualizing turbulent 3D features. *IEEE TVCG 3*, 2 (1997), 129–141. 2

[SWC*08] SCHNEIDER D., WIEBEL A., CARR H., HLAWITSCHKA M., SCHEUERMANN G.: Interactive comparison of scalar fields based on largest contours with applications to flow visualization. *IEEE TVCG 14*, 6 (2008), 1475–1482. 2

[SZM*14] SHIH M., ZHANG Y., MA K.-L., SITARAMAN J., MAVRIPLIS D.: Out-of-core visualization of time-varying hybrid-grid volume data. In *Large Data Analysis and Visualization, 2014 IEEE 4th Symposium on* (2014), pp. 93–100. 2

[TLS12] TONG X., LEE T.-Y., SHEN H.-W.: Salient time steps selection from large scale time-varying data sets with dynamic time warping. In *Large Data Analysis and Visualization (LDAV), 2012 IEEE Symposium on* (2012), pp. 49–56. 2

[vPOBB*11] VAN PELT R., OLIVAN BESCOS J., BREEUWER M., CLOUGH R., GROLLER M., TER HAAR ROMENY B., VILANOVA A.: Interactive virtual probing of 4D MRI blood-flow. *IEEE TVCG 17*, 12 (2011), 2153–2162. 2

[VWJ*13] VELTEN A., WU D., JARABO A., MASIA B., BARSI C., JOSHI C., LAWSON E., BAWENDI M., GUTIERREZ D., RASKAR R.: Femto-photography: Capturing and visualizing the propagation of light. *ACM Trans. Graph. 32*, 4 (2013), 44:1–44:8. 3, 7

[WCBP12] WIDANAGAMAACHCHI W., CHRISTENSEN C., BREMER P.-T., PASCUCCI V.: Interactive exploration of large-scale time-varying data using dynamic tracking graphs. In *Large Data Analysis and Visualization (LDAV), 2012 IEEE Symposium on* (2012), pp. 9–17. 2

[WDC*07] WEBER G., DILLARD S., CARR H., PASCUCCI V., HAMANN B.: Topology-controlled volume rendering. *IEEE TVCG 13*, 2 (2007), 330–341. 2

[WS03] WOODRING J., SHEN H.-W.: Chronovolumes: A direct rendering technique for visualizing time-varying data. In *Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume Graphics* (New York, NY, USA, 2003), VG '03, ACM, pp. 27–34. 2, 3, 6, 10

[WS09] WOODRING J., SHEN H.-W.: Semi-automatic time-series transfer functions via temporal clustering and sequencing. *Computer Graphics Forum 28*, 3 (2009), 791–798. 2

[WWS03] WOODRING J., WANG C., SHEN H.-W.: High dimensional direct rendering of time-varying volumetric data. In *Visualization, 2003. VIS 2003. IEEE* (2003), pp. 417–424. 2

[WYM08] WANG C., YU H., MA K.-L.: Importance-driven time-varying data visualization. *IEEE TVCG 14*, 6 (2008), 1547–1554. 2