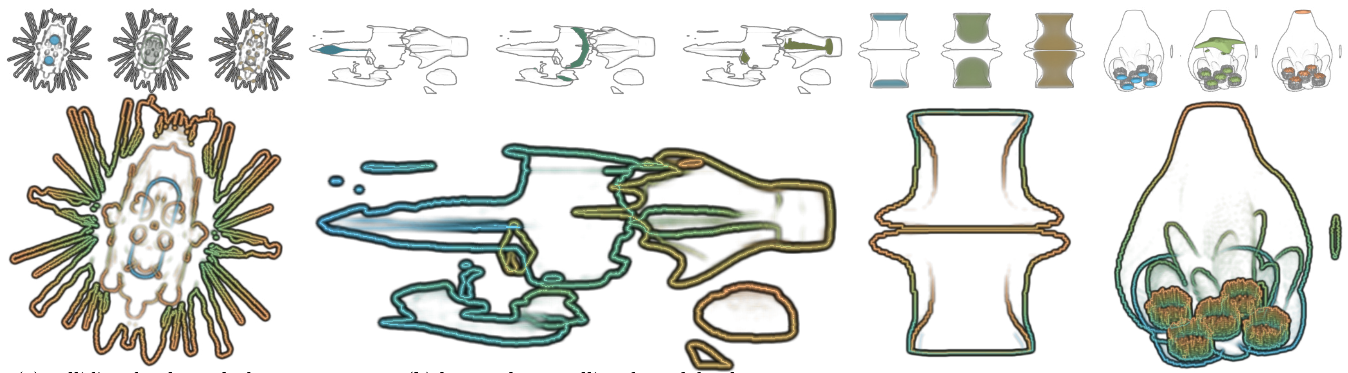
Eurographics Conference on Visualization (EuroVis) 2018
J. Heer, H. Leitte, and T. Ropinski
(Guest Editors)

Volume 37 (2018), Number 3

Spatio-Temporal Contours from Deep Volume Raycasting

S. Frey

University of Stuttgart, Visualization Research Center, Germany



(a) *colliding droplets splash out* **(b)** *laser pulse travelling through bottle* **(c)** *colliding hot and cold air* **(d)** *five jets induce a wave*

Figure 1: Bottom row: *Examples of our contours visualizing bounds and non-continuous changes of spatio-temporal processes. Contour color indicates temporal occurrence (begin ⊣ 🟦🟦🟦🟩🟩🟨🟨🟧🟧 ⊢ end).* Top row: *Renderings of individual time steps roughly outline temporal development, with gray contours in the background for context (cf. Sec. 7 for a detailed discussion of individual examples).*

Abstract

We visualize contours for spatio-temporal processes to indicate where and when non-continuous changes occur or spatial bounds are encountered. All time steps are comprised densely in one visualization, with contours allowing to efficiently analyze processes in the data even in case of spatial or temporal overlap. Contours are determined on the basis of deep raycasting that collects samples across time and depth along each ray. For each sample along a ray, its closest neighbors from adjacent rays are identified, considering time, depth, and value in the process. Large distances are represented as contours in image space, using color to indicate temporal occurrence. This contour representation can easily be combined with volume rendering-based techniques, providing both full spatial detail for individual time steps and an outline of the whole time series in one view. Our view-dependent technique supports efficient progressive computation, and requires no prior assumptions regarding the shape or nature of processes in the data. We discuss and demonstrate the performance and utility of our approach via a variety of data sets, comparison and combination with an alternative technique, and feedback by a domain scientist.

CCS Concepts

•***Human-centered computing*** → ***Visualization techniques; Scientific visualization;***

1. Introduction

Increasingly fast computing systems for simulations as well as high-accuracy measurement techniques enable the generation of time-dependent data sets with high resolution in both time and space. To analyze this kind of data, the most popular approach is to manually browse through the time steps individually or to investigate respective animations. This allows to look at the full spatial information, and well-known rendering and interaction techniques can be used. However, relying on animation alone has been shown to be ineffective as only a limited number of frames can be memorized by an observer (e.g., [JR05]). Generating a static visualization for spatio-temporal data is challenging though due to issues ranging from occlusion and visual clutter to performance limitations for large data. Mitigating this via temporally sparse techniques selecting only a subset of time steps for visualization (e.g. [LS08, TLS12, FE16]) requires either selection criteria tuned for a specific type of data or involves the costly explicit quantification of time step differences, and interesting process transitions may still be missed. While temporally dense techniques visualize all time steps, these typically restrict themselves to specific (user-defined) characteristics or features to circumvent the occlusion problem ([BVMG08, JR05, LSB*17]).

In this paper, we introduce a temporally dense technique (comprising all time steps) that is generic (requires no assumption about the shape or nature of processes in the data), and can be computed efficiently even for larger data. Our contour visualization technique is targeted towards indicating where and when non-continuous changes occur, and with this depict spatio-temporal processes in the data both spatially and temporally (e.g., Fig. 1). Our spatially sparse visualization can further be complemented with volume rendering-based techniques to also show the spatial structure of selected time steps. In the following, after reviewing related work in Sec. 2, we present the main contributions of this work:

- our progressive visualization approach using spatio-temporal contours (Sec. 3), comprised of individual components for . . .
- . . . deep raycasting collecting samples (Sec. 4),
- . . . distance quantification for space-time-value samples (Sec. 5),
- . . . and contour rendering (Sec. 6);
- a discussion of results with different data sets, a time step selection technique [FE16], and domain scientist feedback (Sec. 7).

We finally conclude our work in Sec. 8.

2. Related Work

Time-varying data visualization. A variety of different approaches has been proposed for the visualization of time-dependent 3D data. One line of work treats the data as a space-time hypercube, and applies extended classic visualization operations like slicing and projection techniques [WWS03] or temporal transfer functions [BVMG08] (cf. Bach et al. [BDA*16] for an overview of techniques in this area). 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 on this basis. Among others, they use the earth mover's distance (EMD, aka the Wasserstein metric), which is a common metric to compute the difference between mass distributions (conceptually, it determines the minimum cost of turning one (mass) distribution into the other) [RTG00]. Based on the concept of the EMD, Frey and Ertl [FE17b, FE17a] presented a technique to generate transformations between arbitrary volumes, providing both expressive distances and smooth interpolates. Among others, they apply it to the streaming selection of time steps in temporal data that allows for the reconstruction of the full sequence with a user-specified error bound. Frey [Fre17] discusses a neural network-based technique to estimate and progressively compute such distance metrics for time series data. Based on a similar distance metric, Frey and Ertl [FE16] presented an approach to adaptively select time steps from time-dependent volume data sets for an integrated visualization, with the selection being based on the principle of optimizing the coverage of the complete data. While characteristic time steps can be put into spatial relation via integrated volume rendering, occlusion can be a larger issue for multiply covered regions, and interesting process transitions may still be missed despite the adaptive selection. Strengths and weaknesses of this approach and the contour-based technique introduced in this paper are complementary, as discussed later in Sec. 3 and demonstrated in Sec. 7.

A large body of work in time-dependent volume visualization is based on feature extraction. Lee et al. [LS09a] extract trend relationships among variables for multi-field time varying data. Time Activity Curves (TAC) that contain each voxel's time series have been used as the basis for different techniques (e.g. [FMHC07, LS09b]). Wang et al. [WYM08] extract a feature histogram per volume block, characterize the local temporal behavior, and classify them via k-means clustering. Based on similarity matrices, Frey et al. [FSE12] detect and explore similarity in the temporal variation of field data. Widanagamaachchi et al. [WCBP12] employ feature tracking graphs. 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 an EMD-based global optimization correspondence algorithm. 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. Tory et al. [TRM*01] compare medical imaging modalities for time-varying medical data as well as several methods for presenting the data (including glyphs) from the view of visualization. Post et al. [PVH*03] and McLoughlin et al. [MLP*10] provide overviews on time-varying flow visualization. Note that the time-dependent volume visualization approaches discussed operate in object space, while we our technique works in image-space.

Illustration-based techniques. The visual style of our contour-based visualization technique bears similarities to different Illustration-based techniques, that often pursue similar goals with different means. Brambilla et al. [BCP*12] give an overview of illustrative techniques for flow visualization. In their categorization, we belong to the low-level visual abstractions, which define how to depict a certain structure (whereas high-level techniques increase the communicative intent of the illustration, like close-ups or exploded views). In contrast to our image-based technique, the vast majority of illustration-based visualization techniques for time-varying data requires explicit higher-level data analysis (like feature tracking), enabling descriptive annotations, but impeding generality. To visualize single volumes, Burns et al. [BKR*05] extract and render sparse linear features, such as silhouettes and suggestive contours. Eden et al. [EBG*07] propose a cartoon rendering style for liquid animations, using bold outlines to emphasize depth discontinuities and oriented textures to convey motion. Based on feature detection and tracking, Joshi and Rheingans [JR05] apply illustration techniques to time-varying data sets, like speedlines, flow ribbons, and strobe silhouettes. In another work, they conducted a user study to demonstrate the effectiveness of illustrative techniques [JR08] (however, only simple movement behavior of a simple square shape was considered). Lu and Shen [LS08] compose sample volume renderings and descriptive geometric primitives into interactive storyboards. Meyer-Spradow et al. [MSRVH06] extract motion dynamics via block matching, and use glyphs and speedlines (among others) for visualization. Balabanian et al. [BVMG08] define a set of temporal styles for temporally overlapping voxels when rendering time-varying volume data. Application-specific techniques have been proposed to visualize the structure and time evolution of hurricanes [JCRS09], and the visualization of ocean eddy characteristics [LSB*17].

3. Overview

Motivation & Concept. The main goal of our visualization is to depict spatio-temporal processes in the data, and locate them both spatially and temporally. For this, we consider discontinuities with respect to space, time, and value. This yields a spatially sparse yet temporally dense overview on the process behavior that efficiently deals with spatial and temporal occlusion (still, overdraw of contours can occur). For instance, in Fig. 1a, details can be seen even though there are several layers of different processes occurring at the center

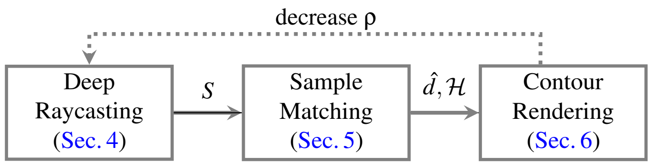


Figure 2: *Overview of spatio-temporal contour visualization: samples S are generated via deep raycasting, and the distances between image cells (captured by a temporal histogram of distances $\mathcal{H}$ and the maximum distance $\hat{d}$) are determined by sample matching for the final contour rendering. For high responsiveness, a progressive approach can incrementally increase the sampling density via ρ.*

at different points in time (initially colliding drops in blue, extending disk in green, and emerging drops in red). Similarly, in Fig. 1d, it can be seen where parts break away and vanish from the big wave going upward, which is not only occluded temporally but also spatially at individual time steps by the wave itself. Conceptually, we resolve the issue of temporal occlusion via abstraction, however, at the cost of neglecting spatial information of individual time steps, which is also crucial for most analysis tasks (particularly when investigating new data for the first time). However, its sparsity allows our approach to be combined with direct volume rendering techniques to supplement spatial information of individual time steps, even in place (e.g., cf. Fig. 4a). In particular, advanced spatio-temporal visualization techniques rendering multiple adaptively selected time steps can be used as well (e.g, [FE16], cf. Fig. 6d, Fig. 7 & Fig. 8f). The combination complements strengths and weaknesses of the individual techniques: while temporal selection-based techniques present spatial information, they strongly suffer from spatial and temporal occlusion, and their temporal sparseness leads to interesting time steps or process transitions being missed, or bounds of the processes not being captured to their full extent. Note that while our technique is primarily designed to provide meaningful static images, it can also be useful to provide temporal context for animations (cf. accompanying videos).

Computing contours across space, time and value requires some kind of normalization to bring these different domains together (in this paper, we use thresholds $\mathcal{T}$ for this). Choosing $\mathcal{T}$ does not strictly require prior information about object shapes or processes in the data, but essentially means steering the sensitivity with which contours are detected (cf. Fig. 3, Fig. 5 & Fig. 8 for respective parameter studies). Finally, note that similar to most other spatio-temporal visualization techniques, we assume that data is spatially registered over time, which is typically inherently true for data from simulations, but may involve extra effort for measured data.

Approach Outline. We give an outline of our progressive approach in Fig. 2. First of all, *deep raycasting* is used to generate a set of samples S_c for each cell (at position c) by following one ray through the center of each cell and then further through the time-dependent data set. To achieve high responsiveness, a progressive approach may be used that incrementally increases the sampling density in image and objects space via resolution parameter ρ. In image space, the standard size of a cell is one pixel, with ρ defining larger (square) side lengths for progressive computation (i.e. $\rho = 4$

Algorithm 1 *Deep raycasting traces rays through space and time and collects space-time-value samples s (cf. Sec. 4). Here, C is the set of cells (in our case, one cell corresponds to a pixel), and V comprises the full view setup (including camera configuration, volume data information, and transfer function). ρ controls the sampling density in time as well as in image and object space.*

```
1: function DEEPRAYCASTING($C, V$)
2:   $S \leftarrow \emptyset$
3:   for all $c \in C_\rho$                          ▷ for all (image space) cells $C$
4:     $S_c \leftarrow \emptyset$
5:     for all $s_{\text{time}} \subset_\rho V_{\text{time}}$          ▷ loop over all considered time steps
6:       ▷ (spatially) sample ray segment in volume bounding box
7:       for all $s_{\text{space}} \subset_\rho \text{IntersectBox}(c, V)^\rho$
8:         $s_{\text{value}} \leftarrow V(c, s_{\text{space}}, s_{\text{time}})$
9:         if $s_{\text{value}} \neq 0$                  ▷ omit full transparent samples
10:          $S_c \leftarrow S_c \cup (s_{\text{space}}, s_{\text{time}}, s_{\text{value}})$
   return $S$
```

results in a 4×4 pixel cell). In contrast to standard volume raycasting, we trace the ray both through space and time, with ρ also directly controlling step sizes in both domains (with the reference for $\rho = 1$ being the sampling once per voxel and time step). No compositing is done but a sample set S is collected and stored for each cell. On this basis, we then compute the distances between cells by looking for nearest neighbors in adjacent cells for each sample (*sample matching*). With this, we determine for each cell the largest distance $\hat{d}$ to a nearest neighbor of any sample, and we create a time histogram $\mathcal{H}$ that captures these distances of all samples. Finally, we use $\hat{d}$ and $\mathcal{H}$ to do *contour rendering*, whereas $\hat{d}$ determines the opacity of cell, and $\mathcal{H}$ is used to compute its color.

4. Deep Raycasting

Deep raycasting generates a set of samples S_c for each cell c by tracing a ray through the center of c through space and time, storing samples instead of compositing them as in standard direct volume rendering (DVR). Below, we introduce space-time-value samples, discuss our deep raycasting procedure conceptually, and finally outline our efficient implementation.

Space-Time-Value Samples. Each sample $s \in S_c$ generated during deep raycasting features one (normalized) scalar for each dimension (value, space, and time):

s_{value} We sample the volume along a ray, fetching the respective data values and directly applying the volume transfer function w.r.t. opacity (making the result consistent with DVR).

s_{time} We further store the normalized time stamp s_{time} of a sample: $s_{\text{time}} = {}^{t - T_{\min}}/_{T_{\max} - T_{\min}}$, with $T_{\min}$ and $T_{\max}$ denoting the first and last time step in the considered time series.

s_{space} Finally, the depth of a sample along a ray in object space is captured. It is quantified via vector projection onto the unit view vector e that goes through the center of the view plane, i.e., $s_{\text{space}} = p \cdot e$, with p being the (normalized) position of the sample s in object space. Note that this yields a maximum value range of $s_{\text{space}} \in [0, \sqrt{3}]$, in contrast to $s_{\text{value}}, s_{\text{time}} \in [0, 1]$.

Deep Raycasting Procedure. In Alg. 1, we conceptually outline

Algorithm 2 *Compute maximum distances $\hat{d}$ and a distance histogram $\mathcal{H}$ w.r.t. time for each cell c (cf. Fig. 5). For this, we match its sample set S_c to the sample set of its right and bottom neighbor ($S_{c\rightarrow}$ and $S_{c\downarrow}$, respectively), considering thresholds $\mathcal{T}$.*

```
1:  function SampleMatching($S, \mathcal{T}$)
2:    for all $S_c \in S$
3:      $\hat{d}_c \leftarrow 0; \mathcal{H}_c \leftarrow 0$
4:      ▷ go right and down bidirectionally
5:      for all $S_a, S_b \in [(S_c, S_{c\rightarrow}), (S_{c\rightarrow}, S_c), (S_c, S_{c\downarrow}), (S_{c\downarrow}, S_c)]$
6:        for all $s_a \in S_a$
7:          $d^* \leftarrow 1$          ▷ initialize distance $d^*$ with 1 (i.e., no match)
8:          for all $s_b \in S_b$          ▷ find the best match in $S_b$ for $s_a$
9:            ▷ distance between samples via $d(\cdot,\cdot,\cdot)$ (Eq. 1)
10:           $d^* \leftarrow \min(d^*, d(s_a, s_b, \mathcal{T}))$
11:         $\mathcal{H}_c \leftarrow_+ (s_a, d^*)$          ▷ update the time histogram
12:         $\hat{d}_c \leftarrow \max(\hat{d}_c, d^*)$          ▷ update match value
    return $\hat{d}, \mathcal{H}$
```

our procedure to generate the set of samples S. For each cell c (Line 3), we trace a ray through time (Line 5) as well as all space (Line 7), and determine the value s_{value} of the sample from the volume (Line 8). The sampling density in time as well as in image and object space is controlled by ρ. Next, we add the sample to the respective set S_c of the cell (Line 10), if the sample value is non-zero (Line 9, avoiding the explicit storage of "empty" space).

Implementation Notes. Deep raycasting potentially creates a vast number of samples, which could induce prohibitively large storage costs for (GPU) memory. For this reason, we densely store samples via a two-pass approach in our implementation (essentially trading computation time for storage efficiency). First, we just count the number of (non-empty) samples generated for each cell, and calculate the offsets for dense sample storage. In the second pass, we then actually collect and store the respective samples (skipping rays that produced no samples in the second pass).

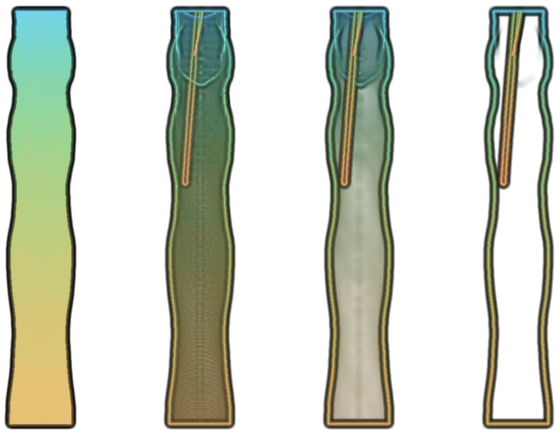
5. Sample Matching for Quantifying Distances between Cells

With the samples S collected via deep raycasting, we now compute distances between samples of neighboring cells. First, we explain how distances between samples are computed, before discussing the steps to determine $\hat{d}$ and $\mathcal{H}$ for each cell.

Distance between Space-Time-Value Samples. To quantify the total distance d between samples, we individually compute distances w.r.t. space, time, and value. For each dimension, we consider a threshold $\mathcal{T}$ that essentially defines until when two samples can still count as a match. Using $\mathcal{T}$, the distance function $d(a,b,\mathcal{T})$ between two samples a and b is computed as follows:

$$d(a,b,\mathcal{T}) = \min(1, \max(||a_{\text{value}} - b_{\text{value}}||/\mathcal{T}_{\text{value}}, \\ ||a_{\text{space}} - b_{\text{space}}||/\mathcal{T}_{\text{space}}, \\ ||a_{\text{time}} - b_{\text{time}}||/\mathcal{T}_{\text{time}})). \quad (1)$$

Here, $d(\cdot,\cdot,\mathcal{T}) = 0$ means that the samples are a close match, while there is no match with $d(\cdot,\cdot,\mathcal{T}) \geq 1$. Here, we only consider the largest occurring individual distance to clearly represent large discontinuities, and to prevent these from getting indistinguishable



(a) $\mathcal{T}_{time} = 0$. **(b)** 0.0125 **(c)** 0.025 **(d)** 0.1

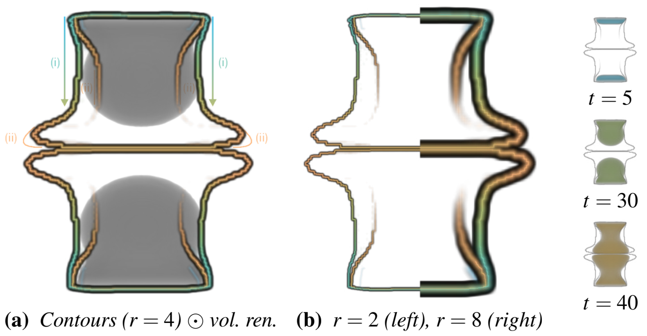
Figure 3: *Impact of $\mathcal{T}_{time}$ on the contours of the Droplet Measurement data, an experiment of a droplet forming at the top and eventually falling to the bottom. (a) For $\mathcal{T}_{time} = 0$ all (non-empty) samples contribute. (b) For a small threshold ($\mathcal{T}_{time} = 0.0125$), the second smaller drop coming down later can now be seen, and it shows that the initial large drop takes some time to form in the top before falling down. Further increasing the threshold to (c) $\mathcal{T}_{time} = 0.025$ and (d) $\mathcal{T}_{time} = 0.1$ reduces visual clutter induced by limited spatial and temporal resolution, but also loses details.*

in the case of overlap with other processes (as would be the case when averaging distances, for example). Essentially, $\mathcal{T}$ defines the sensitivity with which contours are detected. We use $\mathcal{T}_{\text{space}} = 0.4$ and $\mathcal{T}_{\text{value}} = 0.5$ unless otherwise noted ($\mathcal{T}_{\text{time}}$ is reported for each case as it also depends on temporal data resolution).

Distance Computation Procedure. We individually match the sample set S_c of each cell c against the sample sets of its neighbors to the right $S_{c\rightarrow}$ and below $S_{c\downarrow}$ (Alg. 2). For each neighboring set, we first attempt to find the best match for all samples ($\in S_c$) in the other set ($S_{c\rightarrow}$ or $S_{c\downarrow}$), and then the other way around (Line 5). In detail, for each sample in the one set $s_a \in S_a$ (Line 6), we aim to determine the smallest distance d^* to any $s_b \in S_b$, i.e., we compute $d^* \leftarrow \min_{s_b \in S_b}(d(s_a, s_b, \mathcal{T}))$ (Lines 8–10). With this, we update the time histogram $\mathcal{H}_c$ (Line 11) and the maximum distance $\hat{d}_c$ (Line 12) for the cell accordingly (these are used for contour rendering, cf. Sec. 6).

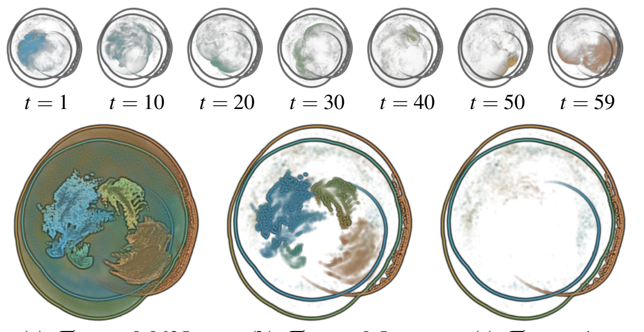
With this procedure, all pairs of cells sharing an edge are considered exactly once. Note that the main focus of interest for our contour visualization lies on differences between the sample sets of cells rather than on the samples of cells by themselves. The purpose of essentially storing the distances of two cell pairs (neighbors to the right and below) in one cell is that is allows for a direct presentation of the respective results (cf. Sec. 6).

Efficient Implementation of Distance Computation. To consider only as few samples as possible when determining the best matching sample $s_b \in S_b$ for s_a (Alg. 2, Line 8), we exit the search early when a s_b with a distance $d^* < 0.1$ has been identified, as these values only induce a negligible impact with low opacity in the contour visualization already. We also generate samples in deep raycasting such that they are implicitly sorted w.r.t. time (cf. Sec. 4, Alg. 1), which means that only samples $s_b \in S_b$ with



(a) *Contours ($r = 4$) ⊙ vol. ren.* **(b)** *$r = 2$ (left), $r = 8$ (right)*

Figure 4: *A hot and cold plate induce collisions of hot and cold air in the space between them ($\mathcal{T}_{time} = 0.2$). (a) When cold air moves down, it maintains a relatively broad shape (i) (vice versa for hot air moving upward). During the course of their collision, their shape becomes thinner closer to their respective plates, but they get broader in the middle where they meet (ii). (b) We also compare the impact of different splat radii r.*



(a) $\mathcal{T}_{value} = 0.0625$ **(b)** $\mathcal{T}_{value} = 0.5$ **(c)** $\mathcal{T}_{value} = 4$

Figure 5: *Simulation of a supernova, in which the supernova constantly rotates throughout the course of the time series. A large temporal threshold $\mathcal{T}_{time} = 0.4$ is used, and the spatial threshold is varied to indicate inner processes (a, b), or exclusively focus on the outline of the rotation (c).*

${s_b}^{\text{time}} \in [{s_a}^{\text{time}} - \mathcal{T}_{\text{time}}, {s_a}^{\text{time}} + \mathcal{T}_{\text{time}}]$ need to be taken into account. In addition, we start the search from the best matching sample ${s_b}^*$ for the previously considered s_a (i.e., from the previous iteration of the loop from Line 6), as often close samples along a ray produce good matches due to coherence. From ${s_b}^*$ we simultaneously search in both directions of the sample list until aforementioned range is left or a distance d^* low enough for an early exit is identified.

6. Contour Rendering

We now determine the color $\mathcal{C}$ of each cell c with the maximum distance value $\hat{d}$ of any of its samples S_c to its right and the bottom neighbors ($S_{c\rightarrow}$ and $S_{c\downarrow}$), and its histogram $\mathcal{H}$ of difference values with respect to time. Here, $\hat{d}$ maps to opacity $\mathcal{C}_\alpha$, while $\mathcal{H}$ determines to RGB color $\mathcal{C}_{\text{rgb}}$. Opacity $\mathcal{C}_\alpha$ is determined directly based on $\hat{d}$ via $\mathcal{C}_\alpha = \hat{d}^2$ (note that $\hat{d} \in [0,1]$). This emphasizes areas with large distances and suppresses the small changes that naturally occur whenever the data undergoes transformation processes. To determine $\mathcal{C}_{\text{rgb}}$, we use the (normalized) histogram $\mathcal{H}$ to compute a weighted sum of our color map. For this, we use a isoluminant color map [Kov15] (i.e., featuring colors of equal perceptual lightness) to achieve good visibility of all time steps on a white background. To further improve the readability of contours, we splat $\mathcal{C}$ with a Gaussian kernel on our image to get thicker lines, and blend color to black along the outside. The respective weights w are computed via $w = \exp(x^2/(2r^2))$, with distance x to the respective pixel and a radius of $r = 4$ (if not specified otherwise). This weight w is then used to blend the color toward black (via $w\mathcal{C}_{\text{rgb}}$), and to decrease the visual impact via the opacity ($w\mathcal{C}_\alpha$). The result is then simply composited with the current image pixel via the over-operator ⊙. Splatting is implemented as a gather and not a scatter operation to allow for efficient parallel execution without atomics. Finally, the resulting contour visualization may be presented to the user on its own or composited with a volume rendering.

7. Results

In our evaluation, we use the data sets listed in Tab. 1 (also see accompanying videos), and an image resolution of 1024×1024. We first discuss the expressiveness of our contour visualization (Sec. 7.1), and report the feedback of a domain scientist regarding the Droplet Simulation (Sec. 7.2). Afterwards, we outline the implementation of our contour visualization system (Sec. 7.3), before evaluating its performance (Sec. 7.4). Finally, we discuss limitations along with potential directions to address them in future work (Sec. 7.5).

7.1. Spatio-Temporal Contour Visualization

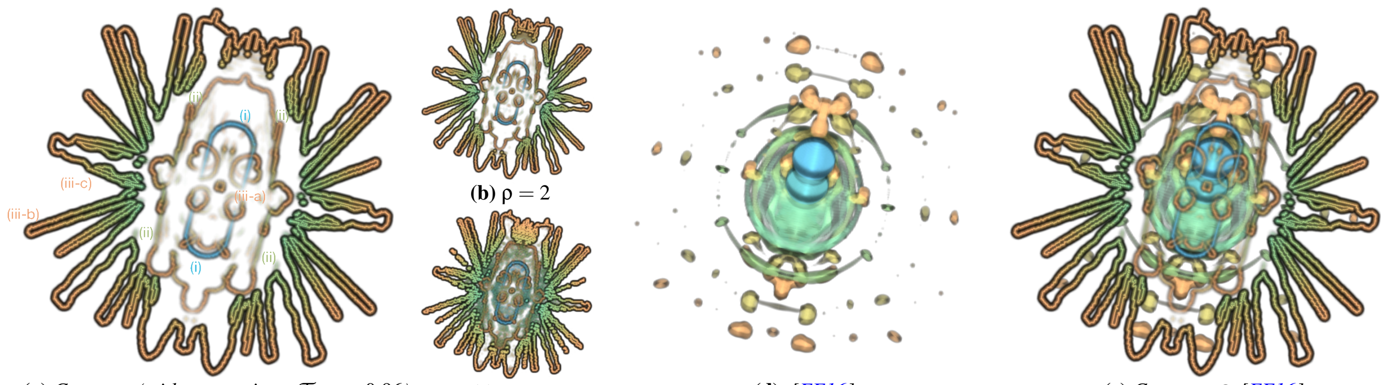
Below, we first outline spatio-temporal bounds for comparably simple processes ***(a)***, before addressing the visualization of more complex, temporally overlapping spatio-temporal processes ***(b)***. After that, we evaluate the case of recurrent processes ***(c)***, and discuss the impact of thresholds $\mathcal{T}$ as well as the splatting radius r ***(d)***. Finally, we compare our contour visualization results to the temporal selection-based technique by Frey and Ertl [FE16], and discuss the results of combining them into one visualization ***(e)***.

(a) Spatio-Temporal Bounds

Here, we choose comparably large values for the temporal threshold $\mathcal{T}_{\text{time}}$ to get a rough overview of the process behavior (yet neglecting details). In the **HotRoom** data set (Fig. 4), a hot and cold plate lead to collisions of hot and cold air in the space between them. The contours indicate that when cold air moves down (or vice versa the body of hot air moves up), it exhibits a relatively broad shape of approximately the size of the plates (Fig. 4a, (i)). It also shows that in the course of the collision, the body of hot and cold air becomes thinner toward their respective plates, but they extend in the middle where they meet (ii). The **Supernova** data (Fig. 5) depicts the results of a supernova simulation. The contours in Fig. 5c primarily depict the outer bound of structure evolving over time, indicating a rotating movement of the supernova (for lower value thresholds $\mathcal{T}_{\text{value}}$ in Fig. 5a & b more details are visible, cf. discussion below in ***(d)***).

(b) Complex, Temporally Overlapping Processes

Next, we analyze more complex processes, also with the goal to



(a) *Contours (with annotations,* $\mathcal{T}_{time} = 0.96$*)* **(b)** $\rho = 2$ **(c)** $\rho = 4$ **(d)** *[FE16]* **(e)** *Contours* $\odot$ *[FE16]*

Figure 6: *Two drops fly toward each other (i), collide asymmetrically, and form a disk from which then individual droplets emerge again (ii) (a). The splash out movement for droplets closer to the center of the disk (iii-a) is much smaller compared to droplets at the vicinity of the disk (iii-b). For the larger droplets, their contours indicate constant changes due to droplet oscillation, e.g., (iii-c). We also show the results for different resolution parameters* ρ *((b) & (c)) as well as respective results of the temporal selection technique by Frey and Ertl [FE16], both by itself (d) and in combination with our contour-based visualization (e).*

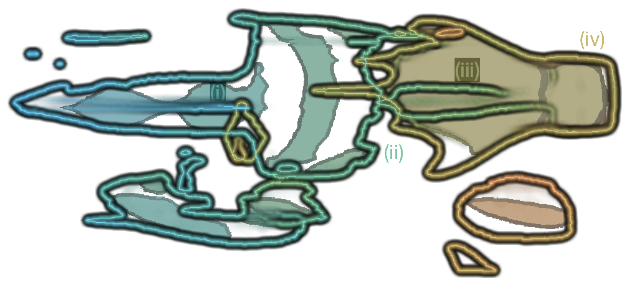


Figure 7: *The* ***Bottle*** *data set depicts a laser pulse traveling through a bottle (captured via Femto Photography [VWJ*13]). As indicated by the contours (*$\mathcal{T}_{time} = 0.06$*), on its way through the bottle from left to right, the laser gets reflected and scattered in different ways with sharp process transitions. Here, the contour image is complemented with the temporal selection of Frey and Ertl [FE16] with five selected time steps (cf. Fig. 1b for just our contours).*

able to see and analyze different phases. For this, we particularly use relatively small values for $\mathcal{T}_{\text{time}}$ to maintain finer details, although at the cost of a higher visual complexity.

The **Droplet Simulation** data set is a simulation of a droplet collision that can roughly be separated in different phases (Fig. 6a). Initially, two droplets move toward each other until they collide asymmetrically. This is indicated by the blue outline in the center (annotated via (i)). After this, the two droplets merge and form a disk from which then individual droplets emerge again (ii). While the disk extends in size, different parts get separated, until eventually individual droplets constitute themselves. This happens both toward the inner parts as well as the outer boundaries of the disk. The individual droplets then continue flying outward, as can be nicely seen from the contours. In addition, it also shows that the splash out movement for droplets toward the center of the disk (iii-a) is much smaller compared to droplets at the vicinity of the disk (iii-b). For the larger droplets, the variation in their contours further indicates constant changes during the outward movement (this is due to droplet oscillation [KBE*17], e.g., (iii-c)). We also provide domain expert feedback for this data set below in Sec. 7.2.

The **Bottle** data set depicts a laser pulse shooting through a bottle (Fig. 1b, Fig. 7). Our contour visualization clearly shows the overall movement of the laser pulse from left to right, as well as several sharp transitions between different process phases. Individual spots around the bottle further depict the appearance of reflecting light in the surrounding. Early on, the pulse forms a structure with a sharp tip (Fig. 7, (i)). Later, it smoothly transitions into a wave with a large vertical extent that ends as indicated by a green line in the contour visualization (ii). Finally, a small laser pulse reappears that initially moves left-to-right (iii) and eventually fills the top of the bottle (iv).

The **5Jets** data set results from a simulation modeling five jets entering a region (Fig. 8). They induce an upward movement process, splitting early into waves of different speeds. Most prominently via its large "bell" shape, the visualization in Fig. 8c shows the movement of the main wave going from the bottom to the top, with its extent decreasing in the process. The blue curved lines in the center indicate the point in time and the position when and where a larger chunk of the big wave vanishes (cf. Fig. 8 *(left column)*, $t = 97$). The green lines close to the jets indicate smaller parts that break away from the bottom of the big wave, and then vanish halfway to the top ($t = 162$). Here, our contour visualization shows where these waves disappear, although this would be occluded in a volume rendering. The same is true for a second phase of smaller waves breaking away (around $t = 208$, just below the main wave). In both phases of waves breaking away, the almost uniform color of the lines indicates that these move relatively quickly, at approximately the same speed of the main wave. The densely drawn contours at the bottom, upward from the five jets, indicates the high variation in their vicinity throughout the whole time series.

(c) Analysis of Recurrent Behavior

Next, we want to discuss how recurrent behavior manifests itself

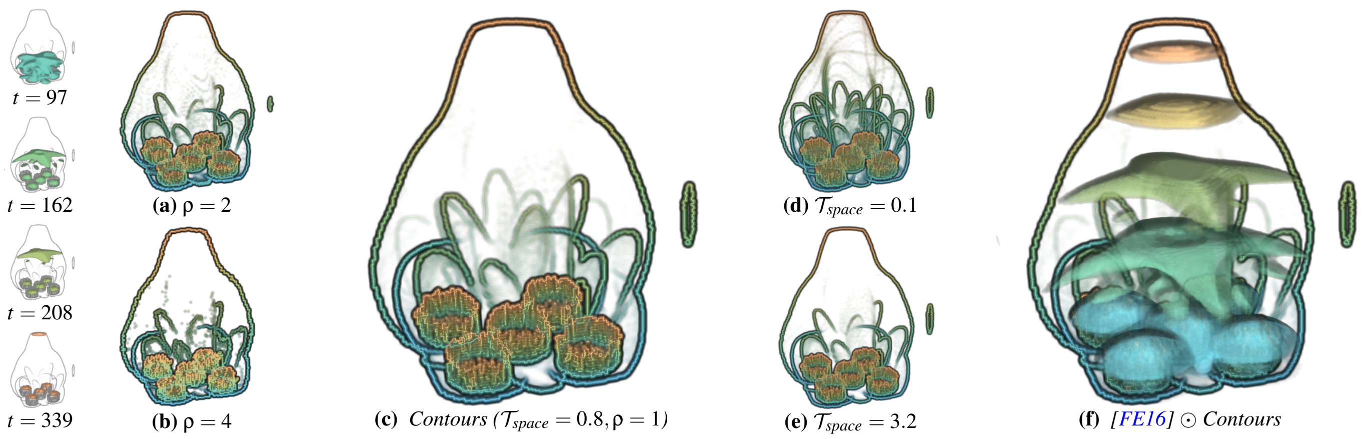

$t = 97$
$t = 162$
$t = 208$
$t = 339$
(a) $\rho = 2$
(b) $\rho = 4$
(c) *Contours* $(\mathcal{T}_{space} = 0.8, \rho = 1)$
(d) $\mathcal{T}_{space} = 0.1$
(e) $\mathcal{T}_{space} = 3.2$
(f) *[FE16]* $\odot$ *Contours*

Figure 8: *The **5Jets** data set is the result of a simulation modeling five jets. It depicts an upward movement which splits into waves of different speeds (with a large wave going all the way to the top). The dense color transition at the five jets at the bottom indicates steadily high variation in the upward movement in their vicinity, while the blue and green "curves" indicate small, quickly moving waves that also terminate quickly. With the contour image acting as a reference (c), we also show the results for different resolution parameters* ρ *((a) & (b)), different spatial thresholds* $\mathcal{T}_{space}$ *((d) & (e)), as well as a combination with the temporal selection technique by Frey and Ertl [FE16] (f).*

with our contour visualization at the example of the **von Kármán** data set (Fig. 9). First, we look at a contour visualization that shows the full time range (Fig. 9a). The visualization essentially shows different stages of development. In the beginning, the process develops mainly in horizontal direction, with smaller left-to-right swings (Fig. 9c & Fig. 9d). After that, there is a transition to recurrent behavior in which mass partitions move from the top to the bottom in essentially two lanes (Fig. 9g & Fig. 9h). In the transition period toward this recurrent behavior, the mass partitions don't go all the way to the bottom but stop earlier (this shows via the green "lines" within each lane, cf. Fig. 9e & Fig. 9f). The two lanes mentioned above appear more clearly in the contour visualization that only depicts the second half of the time series (Fig. 9b). Here, it can be seen that there are no contours within lanes, which indicates that the contained processes exhibit smooth transitions.

(d) Impact of thresholds ($\mathcal{T}_{time}$, $\mathcal{T}_{space}$, $\mathcal{T}_{value}$) & splat radius r

First of all, if any threshold is set to $\mathcal{T} = 0$, only the outer bounds of processes can be seen, as there are no valid matches for any sample (cf. Fig. 3a). Accordingly, with $\mathcal{T} = 0$, essentially all time steps are simply fully composited on top of each other. This shows a rough outline of overall behavior, yet particularly details are lost within regions where multiple processes overlap spatially at different points in time. In general, the larger a threshold $\mathcal{T}$ is, the lower is its sensitivity to changes in the respective domain.

For a small **temporal threshold** ($\mathcal{T}_{\text{time}} = 0.0125$), different structures can be seen that were obfuscated before, i.e., the second smaller drop coming down at a later point in time (Fig. 3b). It can also be seen that the initial drop takes some time to form in the top before falling down. Further increasing the threshold ($\mathcal{T}_{\text{time}} = 0.025$) reduces the contours on the way that were primarily detected due to limitations w.r.t. spatial and temporal resolution (Fig. 3c). An even larger threshold ($\mathcal{T}_{\text{time}} = 0.1$) reduces these resolution artifacts, but also information is lost, e.g., regarding the slowly forming droplet on the top (Fig. 3d). Similarly, a small **value threshold** $\mathcal{T}_{\textbf{value}}$ results in a significant visual impact even of smaller changes in the rotating supernova ($\mathcal{T}_{\text{value}} = 0.0625$, Fig. 5a), while a larger $\mathcal{T}_{\text{value}} = 0.5$ leads to only the larger value changes within the supernova being visible (Fig. 5b). When further increasing the threshold, only the outline of the rotating behavior can be seen ($\mathcal{T}_{\text{value}} = 4$, Fig. 5c). The same behavior can be observed in the spatial domain with the **spatial threshold** $\mathcal{T}_{\textbf{space}}$: lots of smaller details induced by smaller discontinuities can be seen for a small $\mathcal{T}_{\text{space}} = 0.1$ (Fig. 8d), with the sensitivity decreasing for larger threshold values $\mathcal{T}_{\text{space}} = 0.8$ (Fig. 8c) and $\mathcal{T}_{\text{space}} = 3.2$ (Fig. 8e). As can be seen across all sample dimensions, changes to the thresholds $\mathcal{T}$ mostly have an impact on the sensitivity, while the most prominent process outlines persist. The impact of the **splat radius** r behaves as can be expected intuitively: larger values for r mean that contours are more pronounced, however, at the cost of covering more image space (Fig. 4).

(e) Selection-based Visualization: Comparison & Integration

We now compare our contour visualization with a time step selection-based technique at the example of the approach by Frey and Ertl [FE16] (note that we focus on spatial aspects in this work, and therefore use their integrated volume rendering, but their abstract visualization of time step similarity is not further considered in this context). In general, these type of techniques maintain the full spatial information, and intuitively provide a good overview on the processes in the data (e.g., Fig. 6d). However, they suffer from both spatial and particularly temporal occlusion, and their temporal sparseness means that interesting processes or events may be missed, and that the transition between selected time steps is not represented. In contrast, our contour-based approach largely neglects shape information. To alleviate the individual weaknesses and complement each others strengths, we can exploit the sparsity of our contour-based visualization by combining both visualization techniques in one rendering. As discussed in the following, while this comes at the cost of a higher visual complexity, it also significantly increases the amount of information in a single image.

In the volume rendering alone, even though only using a subset of data mitigates temporal occlusion, different selected time steps occlude each other, like in the bottom at the five jets in Fig. 8f and in the center around the droplet collision in Fig. 6e. In addition, in Fig. 8f, there is also spatial occlusion as the waves cover the small elements that split off. In combination with our contour-based technique, we can see where and when individual split-offs appear and vanish. In Fig. 6e, contours indicate where individual droplets split of at the outer disk, and supplement additional information what happens in the center area that suffers from strong temporal overlap. In addition, the bounds of the processes cannot be assessed completely from the volume rendering alone, but are supplemented via contours, like how far the disk and the droplets splat out (Fig. 6e), or how long individual parts persist and how far the wave extends upwards (Fig. 8f). Our contour-based visualization further adds missing information regarding process transitions, like the fact that and when individual droplets smoothly form from the splashing out disk (Fig. 6e), or when and where sharp transitions occur for the pulse traveling through the bottle (Fig. 7)

Similar to our technique (see performance discussion below in Sec. 7.4), the selection-based technique by Frey and Ertl [FE16] also supports a progressive approach, yet the costs overall are significantly higher: $|T|/2$ time steps need to be compared to complete the refinement (with each comparison being in the order of hundreds of ms), and the selection of time steps requires the adequate exploration of the $\binom{|T|}{|S|}$ possibilities to select $|S|$ time steps from $|T|$ time steps. For the 5Jets and the Droplet simulation data sets, first selection results where generated after 61 s and 83 s, with 4733 s and 13 096 s until full completion, respectively (cf. Frey and Ertl [FE16] for a detailed performance analysis).

7.2. Domain Expert Feedback

We presented our approach to an expert from the field of aerospace thermodynamics regarding the Droplet Simulation data. His feedback is reported in the following.

Drop collisions appear in many technical applications, like fuel injection, fire suppression and spray drying, as well as in nature, for example in clouds, and are therefore a relevant topic in multiphase flows. Numerical studies are often used, to investigate the influence of different parameter, such as the relative velocity of the drops or the excentricity of the collision. When investigating large sets of parameters it becomes important, to be able to quickly understand the major differences in the simulation results. As droplet collisions are a highly transient process, which needs to be highly resolved in time, obtaining a quick overview has proven difficult.

The proposed method, applied to a DNS of a drop collision gives insight into several relevant aspects of the physical process. The amount of droplets ejected from the center is represented by the amount of finger-like structures. These structures also provide the trajectories of the droplets, as well as an approximate information on their velocity, given by the distance to the collision. The width of the fingers provides information of the size of the droplets. Furthermore, the wave-like structure of the contour represents droplet oscillations and can even provide some insight into the frequencies, amplitudes and dampening of the droplets in relation to other droplets. It can also be observed that the smaller droplets do not exhibit oscillating behavior, which due to their lower liquid mass are below the critical Ohnesorge number. The presented method provides a fast way to gain a broad qualitative insight into different temporal and spatial aspects of a drop collision and would be well-suited to support numerical parameter studies of this phenomenon.

7.3. Implementation of the Contour Visualization System

While volume rendering of individual time steps is typically sufficiently fast to deliver interactive performance, deep raycasting and difference computations for contour generation induce significantly higher costs (cf. discussion below in Sec. 7.4). Not only is the complete data set (space and time) considered for deep raycasting, but also a matching between different sample sets needs to be computed for each cell. These costs essentially prohibit our approach to run at full resolution with interactive rates for user exploration (at least with our current implementation). While it is not our primary goal in this work to create an interactive visualization system with our contours, we still try to minimize wait times (cf. our discussion regarding the efficient implementation of individual components in Sec. 4 and Sec. 5), and integrate our space-time contours into a system that shows computed results when become available.

To yield a system that is responsive at any time, we run contour computations in a separate thread from volume rendering and the main interaction thread. Here, we take a progressive approach that quickly generates low-resolution results, which are then replaced by higher-resolution renderings as soon as they become available. In our implementation, we successively compute three different resolution levels: $\rho = 4$, $\rho = 2$, and finally $\rho = 1$ (which corresponds to the full resolution in image and object space, as well as time). Whenever a user interaction necessitates a new computation run, we notify the contour computation thread such that it aborts it current procedure and restarts to complete the new request. This occurs whenever the camera or the transfer function are changed (which triggers a full recomputation), or the thresholds $\mathcal{T}$ are adjusted (only the matching and rendering need to be updated). To be able to compute new results in the background, we use two buffers storing the visualization-relevant data, which are swapped whenever a new result becomes available (naturally, old results are not displayed anymore once they have been invalidated). When two GPUs are available, we dedicate the faster GPU to deep raycasting and difference computation, while the other one handles user interaction and standard raycasting.

OpenGL and GLSL are used to render the volume. We designed the time-relevant steps of contour visualization (deep raycasting, difference computation, and contour rendering) to be easy to parallelize and implemented them using CUDA. All cells can be processed independently throughout our pipeline, and we run respective computations in parallel: each ray in deep raycasting, the matching procedure for each cell for distance computation, and contour splatting for rendering. In deep raycasting, we use nearest neighbor lookups to avoid introducing artificial values through interpolation.

7.4. Performance Evaluation

We evaluate the implementation of our approach on a machine equipped with an Intel Core i7-6700 CPU, 32 GB of memory, and two NVIDIA GPUs. We use a GeForce GTX 1070 (with 8 GB of memory) for the interactive display via OpenGL (including standard

Data Set **Name**, spatial resolution, time steps	DR pass 0 (in s) (Sec. 4)	DR pass 1 (in s) (Sec. 4)	$\lvert S\rvert$ (in Millions) number of samples	Distances (in s) (Sec. 5)	Rendering (in s) (Sec. 6)	Total (in s) $\rho = 1$ $\rho = 2$ $\rho = 4$
5Jets (Fig. 8), $128^3, 340$	8.28 0.54 0.04	5.23 0.35 0.03	338.12 20.84 1.26	10.83 0.52 0.06	0.129 0.129 0.129	24.47 1.54 0.26
Bottle (Fig. 7), $900 \times 430, 300$	0.06 0.01 0.00	0.03 0.00 0.00	4.58 0.57 0.07	0.03 0.00 0.00	0.151 0.136 0.129	0.27 0.15 0.13
Droplet Measurement (Fig. 3), $182 \times 878, 315$	0.02 0.00 0.00	0.02 0.01 0.00	2.09 0.26 0.03	0.01 0.00 0.00	0.129 0.129 0.129	0.18 0.14 0.13
Droplet Simulation (Fig. 6), $256^3, 500$	39.04 2.63 0.22	21.69 1.59 0.13	143.75 8.98 0.56	7.46 0.58 0.09	0.132 0.130 0.130	68.32 4.93 0.58
Hotroom (Fig. 4), $181 \times 91^2, 50$	1.58 0.10 0.01	0.70 0.05 0.00	210.87 12.89 0.71	5.24 0.25 0.02	0.129 0.129 0.129	7.65 0.53 0.16
von Kármán (Fig. 9), $101 \times 301, 800$	0.11 0.02 0.00	0.05 0.01 0.00	14.82 1.85 0.23	0.19 0.03 0.02	0.130 0.129 0.161	0.48 0.18 0.18
Supernova (Fig. 5), $432^2, 60$	8.05 0.55 0.05	6.25 0.43 0.03	451.15 28.16 1.75	11.93 0.60 0.05	0.131 0.131 0.131	26.36 1.71 0.26

Table 1: *Performance for data sets and resolution levels $\rho \in [1, 2, 4]$ (from left to right in each table cell). DR stands for deep raycasting.*

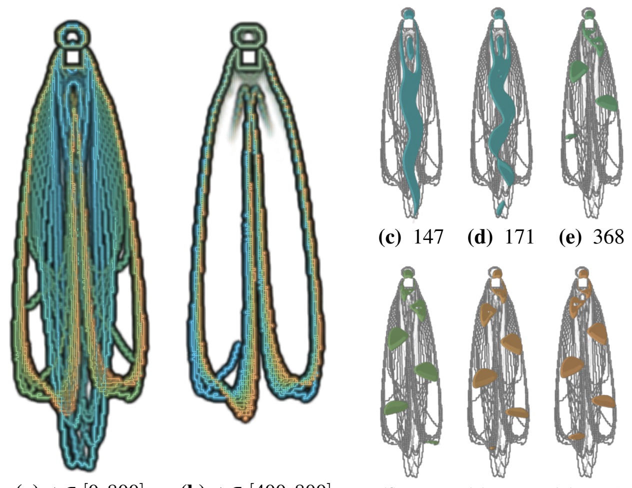


Figure 9: *CFD simulation of a von Kármán vortex street. (a) Different stages of development can be seen ($\mathcal{T}_{time} = 0.04$). In the beginning of the simulation, the process develops mainly horizontally, with smaller scale swings from the left to the right (Fig. 9c & Fig. 9d). After that, there is a transition to recurrent behavior moving from top to bottom in what are essentially two lanes (Fig. 9g & Fig. 9h). (b) This is shown more clearly in the contour visualization only depicting the second half of the time series.*

volume raycasting and rendering of the overlay). For deep raycasting, difference computation, and distribution computation, we use a TITAN X with 12 GB of memory. Tab. 1 shows results for individual steps of our approach across all data sets considered in this paper, as well as different resolution parameters ρ. Regarding deep raycasting, it can be seen that the first raycasting pass (DR pass 0) typically takes longer than the second pass (DR pass 1), despite the fact that the second pass actually needs to store samples. However, we are able to only consider non-empty cells in DR pass 1, which significantly decreases the number of rays that need to be traced. Accordingly, it shows that this difference also depends on the fraction of empty cells in the respective view (cf. renderings). For instance, it is almost a factor of two for the Droplet Simulation, but less for 5Jets and the Supernova. Overall, as our views in all cases practically capture the full volume, these timings are related to the total number of voxels in the data set (in space and time).

The total time for the distance computation is strongly related to the total number of samples in a data set ($|S|$). However, other factors have an impact as well. This can be seen for instance when comparing the Droplet Simulation and the Hotroom. Here, the Hotroom more strongly benefits from our optimizations regarding sample matching coherency (cf. Sec. 5) as it features a massive, coherent body of mass in object space, and therefore yields a faster computation than the Droplet Simulation even though it features more samples in total. The time to actually render the contours is around 20 ms (and hence rather negligible overall). Naturally, there is a huge performance gap between the 2D data sets (Bottle, Droplet Measurement, and von Kármán), which even compute at interactive rates for the full resolution ($\rho = 1$), and the 3D data sets (5Jets, Droplet Simulation, Hotroom, and Supernova) due to the much larger amount of samples that need be generated and processed.

In our progressive approach, we do not generate the final result right away (with resolution parameter $\rho = 1$), but start with larger values to achieve higher responsiveness in interactive applications. For this, we consider two more resolution levels with $\rho = 2$ and $\rho = 4$. Doubling ρ means that only a quarter of the rays are traced, with half the step size in object space and half the time steps, resulting in a theoretical speedup of 16×. However, at runtime the actual performance scaling varies due to a variety of influence factors like reduced coherence, with an overall lower speedup when going from $\rho = 2$ to $\rho = 4$ compared to $\rho = 1 \rightarrow \rho = 2$ (or, in general, when considering cases with lower numbers of samples). As can be seen from Tab. 1, practically, we observe overall speedups for our computationally intense 3D data sets from $\rho = 1$ to $\rho = 2$ of 16.5× for 5Jets, 12.6× for Droplet Simulation, 17.1× for Hotroom, and 13.1× for Supernova. Regarding the visual impact of ρ (Fig. 6a–c & Fig. 8a–c), the visualizations are qualitatively very similar, although the impact of the significantly lower resolution can clearly be seen, especially for $\rho = 4$. In any case, we consider $\rho = 4$ and particularly $\rho = 2$ to be close enough to get a good impression of changes induced by interaction at interactive rates, whether it is changing the camera position or parameters $\mathcal{T}$. However, note that there is still potential for performance improvements with further optimization as discussed below in Sec. 7.5.

7.5. Discussion of Limitations & Directions For Future Work

Expressiveness. Our contour visualization technique is targeted towards indicating where and when non-continuous changes occur or spatial bounds are encountered in the captured spatio-temporal processes. With this, our technique is able to give a comprehensive overview on the whole progression captured in the time-dependent data, but it can also be helpful to investigate local effects. While our approach yields a spatially sparse representation that performs well overall when dealing spatial and temporal occlusion, a large number of processes in the same area can still lead to overdraw and visual clutter. This can be mitigated by reducing the amount of information to be shown, for instance by adjusting the threshold parameters $\mathcal{T}$ to less sensitive (larger) values, or by restricting the considered time frame. In general, not all information that is potentially of interest can be represented by our visualization. Most prominently, our contour visualization does not directly capture the form of the structures in the data, and how their shape changes (as long as changes occur smoothly and continuously). For example, in the Droplet Simulation data, the lanes going outward from the splash only represent the droplet paths, but not the shape of individual droplets themselves. Only if non-continuous changes happen, or a shape persists for a longer period of time, individual shapes can be seen (like the hanging drop in the Droplet Measurement data in Fig. 3). As shown earlier, to overcome this, we can combine our contours with an additional spatial view that can fill in the missing spatial detail for a selected time step. For this, we use standard volume rendering (Fig. 4) and the approach by Frey and Ertl [FE16] (Fig. 7, Fig. 6 & Fig. 8), but other approaches would work as well [LS08, TLS12]. However, if used as an overlay to volume rendering, our contours also occlude a portion of the volume rendering (or vice versa, depending on the order of compositing). Naturally, the issue of overdraw can also exist for contours by themselves, when there are many complex processes occurring in a certain area over the course of a time series. This can be mitigated by carefully choosing parameters, i.e., the fraction of the screen space taken by the contours can be controlled via the thresholds $\mathcal{T}$, and the Gaussian we use for contour splatting could also be adjusted for thicker or thinner contours via r. In addition, different overlay modes can be toggled interactively (compositing order and color / gray).

In our current implementation, the color of the contours only depicts the distribution $\mathcal{H}$ with respect to time. Here, we aim to investigate further visual cues that we could integrate to potentially depict depth or value as well. Regarding interaction possibilities, our contours currently give a good indication of the numerous aspects regarding spatio-temporal behavior, but they do not directly support subsequent closer investigation. For this, we aim to extend the interaction possibilities of our approach to the selection of (4D) subsets of the data by supporting picking on our contours. Furthermore, while our technique conceptually supports data values of arbitrary dimensionality (requiring only a meaningful pairwise distance function), we only considered scalar data in this paper. Accordingly, in future work, we plan to extend and evaluate our approach with higher-dimensional data values like vectors and tensors.

Performance. We further aim to increase the computational efficiency of our approach by improving our implementation in general, and exploring potentials for pre-processing in particular. While our progressive computation scheme adjust the sampling rate via ρ, the actual data is only stored in full resolution. Potential performance issues for large-scale data in particular could be addressed with out-of-core techniques or level-of-detail data representations. In addition, a hybrid CPU-GPU implementation could be employed to fully exploit the amount of available memory on a system. In our current implementation, we extensively use the GPU to achieve low response times. However, for all steps of our approach that run on the GPU, we also have a multicore CPU implementation (we only discussed and evaluated our GPU implementation in this paper due to its superior performance). Finally, we run the deep raycasting procedure twice to achieve dense sample storage. While we already skip rays known to produce no samples the second pass, more elaborate acceleration approaches could be used based on the information from the first pass. We could further develop dynamic storage reservation schemes on the basis of atomic operations to potentially only require a single pass. We also use regular device memory in our implementation, and switching to texture memory for storing volumes in our GPU implementation has the potential to further accelerate deep raycasting.

8. Conclusion

In this paper, we introduced contours for the visualization of spatio-temporal processes. In particular, they indicate where and when non-continuous changes occur or spatial bounds are encountered in the data. All time steps are comprised densely in one visualization, and our contours allow to efficiently analyze processes even when there is spatial or temporal overlap. The contour visualization is based on samples collected via deep raycasting that traces rays across both object space and time to capture spatio-temporal effects. For each sample along a ray, its closest neighbors from adjacent rays are determined, considering time, depth, and value. While we rely on parameters both to combine these different dimensions and to adjust the sensitivity with which the contours depict process discontinuities, we require no prior assumptions regarding the shape or nature of processes in the data. Large distances are represented as contours in image space, mapping color to temporal occurrence. We conceptually discussed and showed that this contour representation can easily be combined with volume rendering-based techniques (including advanced spatio-temporal visualization approaches), providing both full spatial detail for individual time steps and an outline of the whole time series in one view. To allow for a responsive visualization system, our view-dependent technique supports efficient progressive computation. Finally, we discussed and demonstrated the performance and utility of our approach via a variety of data sets, comparison and combination with an alternative technique, and feedback by a domain scientist. For future work, we aim to extend our approach to address limitations and explore new directions (cf. Sec. 7.5 for a comprehensive discussion). Among others, we plan to support higher-dimensional data values like vectors and tensors, further explore the integration with complementary spatio-temporal visualization techniques, and extend interaction possibilities.

Acknowledgments

The authors would like to thank the German Research Foundation (DFG) for supporting the project within project A02 of SFB/Transregio 161, project D01 of SFB 1313, and the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart. The authors would like to thank Moritz Ertl from the Institute for Aerospace Thermodynamics (ITLR) at the University of Stuttgart for providing the dataset for the droplet collision case. The simulations were calculated with the multiphase flow DNS code FS3D [EEG*16]. The dataset was provided under a cooperation as part of the SFB-TRR 75 “Droplet Dynamics Under Extreme Ambient Conditions”. The Supernova data set is made available by Dr. John Blondin [BM07], and the 5Jets data set stems from the volume data repository of UC Davis.

References

[BCP*12] BRAMBILLA A., CARNECKY R., PEIKERT R., VIOLA I., HAUSER H.: Illustrative Flow Visualization: State of the Art, Trends and Challenges. In *Eurographics 2012 - State of the Art Reports* (2012), Cani M.-P., Ganovelli F., (Eds.), The Eurographics Association. doi:10.2312/conf/EG2012/stars/075-094. 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. *Comput. Graph. Forum* (2016). doi:10.1111/cgf.12804. 2

[BKR*05] BURNS M., KLAWE J., RUSINKIEWICZ S., FINKELSTEIN A., DECARLO D.: Line drawings from volume data. *ACM Trans. Graph. 24*, 3 (July 2005), 512–518. doi:10.1145/1073204.1073222. 2

[BM07] BLONDIN J. M., MEZZACAPPA A.: Pulsar spins from an instability in the accretion shock of supernovae. *Nature 445*, 7123 (2007), 58. 10

[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. 1, 2

[EBG*07] EDEN A. M., BARGTEIL A. W., GOKTEKIN T. G., EISINGER S. B., O'BRIEN J. F.: A method for cartoon-style rendering of liquid animations. In *Proceedings of Graphics Interface 2007* (May 2007), pp. 51–55. 2

[EEG*16] EISENSCHMIDT K., ERTL M., GOMAA H., KIEFFER-ROTH C., MEISTER C., RAUSCHENBERGER P., REITZLE M., SCHLOTTKE K., WEIGAND B.: Direct numerical simulations for multiphase flows: An overview of the multiphase code FS3D. *Journal of Applied Mathematics and Computation 272*, 2 (1 2016), 508–517. doi:10.1016/j.amc.2015.05.095. 10

[FE16] FREY S., ERTL T.: Flow-based temporal selection for interactive volume visualization. *Comput. Graph. Forum* (2016). doi:10.1111/cgf.13070. 1, 2, 3, 5, 6, 7, 8, 10

[FE17a] FREY S., ERTL T.: Fast flow-based distance quantification and interpolation for high-resolution density distributions. In *EG 2017 - Short Papers* (2017). 2

[FE17b] FREY S., ERTL T.: Progressive direct volume-to-volume transformation. *IEEE T. Vis. Comput. Gr. 23*, 1 (Jan 2017), 921–930. doi:10.1109/TVCG.2016.2599042. 2

[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. doi:10.1145/1268517.1268563. 2

[Fre17] FREY S.: Sampling and estimation of pairwise similarity in spatio-temporal data based on neural networks. *Informatics 4*, 3 (2017). doi:10.3390/informatics4030027. 2

[FSE12] FREY S., SADLO F., ERTL T.: Visualization of temporal similarity in field data. *IEEE Vis. Comput. Gr. 18* (2012), 2023–2032. doi:10.1109/TVCG.2012.284. 2

[JCRS09] JOSHI A., CABAN J., RHEINGANS P., SPARLING L.: Case study on visualizing hurricanes using illustration-inspired techniques. *IEEE Transactions on Visualization and Computer Graphics 15*, 5 (Sept 2009), 709–718. doi:10.1109/TVCG.2008.105. 2

[JR05] JOSHI A., RHEINGANS P.: Illustration-inspired techniques for visualizing time-varying data. In *Visualization, 2005. VIS 05. IEEE* (2005), pp. 679–686. doi:10.1109/VISUAL.2005.1532857. 1, 2

[JR08] JOSHI A., RHEINGANS P.: Evaluation of illustration-inspired techniques for time-varying data visualization. *Comput. Graph. Forum 27*, 3 (2008), 999–1006. doi:10.1111/j.1467-8659.2008.01235.x. 2

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

[KBE*17] KARCH G., BECK F., ERTL M., MEISTER C., SCHULTE K., WEIGAND B., ERTL T., SADLO F.: Visual analysis of inclusion dynamics in two-phase flow. *IEEE Transactions on Visualization and Computer Graphics PP*, 99 (2017), 1–1. doi:10.1109/TVCG.2017.2692781. 6

[Kov15] KOVESI P.: Good colour maps: How to design them. *CoRR abs/1509.03700* (2015). arXiv:1509.03700. 5

[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. doi:10.1109/PACIFICVIS.2008.4475470. 1, 2, 10

[LS09a] LEE T.-Y., SHEN H.-W.: Visualization and exploration of temporal trend relationships in multivariate time-varying data. *IEEE Vis. Comput. Gr. 15*, 6 (2009), 1359–1366. doi:10.1109/TVCG.2009.200. 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. doi:10.1109/PACIFICVIS.2009.4906831. 2

[LSB*17] LIU L., SILVER D., BEMIS K., KANG D., CURCHITSER E.: Illustrative visualization of mesoscale ocean eddies. *Computer Graphics Forum 36*, 3 (2017), 447–458. doi:10.1111/cgf.13201. 1, 2

[MLP*10] MCLOUGHLIN T., LARAMEE R. S., PEIKERT R., POST F. H., CHEN M.: Over two decades of integration-based, geometric flow visualization. *Comput. Graph. Forum 29*, 6 (2010), 1807–1829. doi:10.1111/j.1467-8659.2010.01650.x. 2

[MSRVH06] MEYER-SPRADOW J., ROPINSKI T., VAHRENHOLD J., HINRICHS K. H.: Illustrating dynamics of time-varying volume datasets in static images. 2

[NTN15] NARAYANAN V., THOMAS D. M., NATARAJAN V.: Distance between extremum graphs. In *IEEE Pacific Visualization Symposium* (2015), pp. 263–270. doi:10.1109/PACIFICVIS.2015.7156386. 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. *Comput. Graph. Forum 22*, 4 (2003), 775–792. doi:10.1111/j.1467-8659.2003.00723.x. 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. doi:10.1023/A:1026543900054. 2

[SW97] SILVER D., WANG X.: Tracking and visualizing turbulent 3D features. *IEEE Vis. Comput. Gr. 3*, 2 (1997), 129–141. doi:10.1109/2945.597796. 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 Vis. Comput. Gr. 14*, 6 (2008), 1475–1482. doi:10.1109/TVCG.2008.143. 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. doi:10.1109/LDAV.2012.6378975. 1, 2, 10

[TRM*01] TORY M., RÖBER N., MÖLLER T., CELLER A., ATKINS M. S.: 4d space-time techniques: A medical imaging case study. In *Proceedings of the Conference on Visualization '01* (Washington, DC, USA, 2001), VIS '01, IEEE Computer Society, pp. 473–476. 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. doi:10.1145/2461912.2461928. 6

[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. doi:10.1109/LDAV.2012.6378962. 2

[WDC*07] WEBER G., DILLARD S., CARR H., PASCUCCI V., HAMANN B.: Topology-controlled volume rendering. *IEEE Vis. Comput. Gr. 13*, 2 (2007), 330–341. doi:http://doi.ieeecomputersociety.org/10.1109/TVCG.2007.47. 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. doi:10.1109/VISUAL.2003.1250402. 2

[WYM08] WANG C., YU H., MA K.-L.: Importance-driven time-varying data visualization. *IEEE Vis. Comput. Gr. 14*, 6 (2008), 1547–1554. doi:10.1109/TVCG.2008.140. 2