DOI: 10.1111/cgf.14092

COMPUTER GRAPHICS *forum*
Volume 00 (2020), number 00 pp. 1–15

Temporally Dense Exploration of Moving and Deforming Shapes

S. Frey

Visualization Research Center, University of Stuttgart, Stuttgart, Germany

Abstract
We present our approach for the dense visualization and temporal exploration of moving and deforming shapes from scientific experiments and simulations. Our image space representation is created by convolving a noise texture along shape contours (akin to LIC). Beyond indicating spatial structure via luminosity, we additionally use colour to depict time or classes of shapes via automatically customized maps. This representation summarizes temporal evolution, and provides the basis for interactive user navigation in the spatial and temporal domain in combination with traditional renderings. Our efficient implementation supports the quick and progressive generation of our representation in parallel as well as adaptive temporal splits to reduce overlap. We discuss and demonstrate the utility of our approach using 2D and 3D scalar fields from experiments and simulations.

Keywords: dense visualization, time-dependent scalar data, temporal exploration, parallel computing

ACM CCS: • Human-centred computing → Visualization techniques;

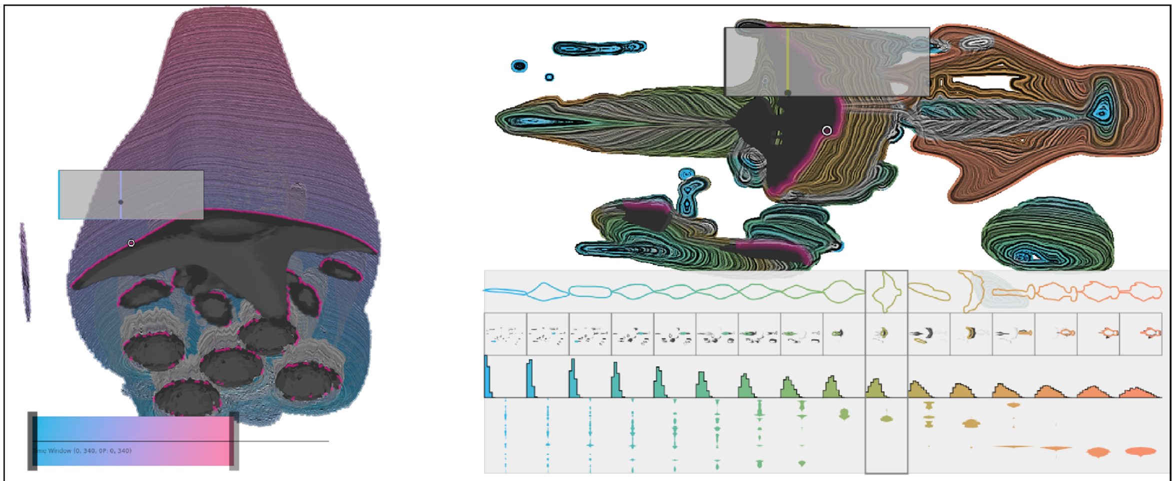
1. Introduction

Increasingly fast computing systems for simulations and high-accuracy measurement techniques enable the generation of time-dependent data sets with high spatial and temporal resolution. In this work, we introduce a visualization approach providing a summary and supporting the exploration of moving and transforming shapes in 2D (e.g. Figure 1b) and 3D scalar field data (e.g. Figure 1a). These occur in various scientific analysis scenarios (e.g. in this work, we will consider moving droplets, laser pulses, waves and areas of high temperature or velocity magnitude). Navigating time via animations has been shown to be ineffective for the analysis as only a limited number of frames can be memorized by the observer (e.g. [JR05]). The typical indirect temporal exploration approach via a time slider is tedious as well, and aspects of interest can easily be missed.

To address this, we present our novel approach for the visualization and navigation of time-dependent field data in image space. Its central component is a new representation that spatially conveys shape information in a temporally dense way (i.e. no temporal selection is required but all time steps in a series are depicted). For this, we employ texture-based visualization that convolves a noise texture along shape contours in a way that is conceptually similar to LIC [CL93]. An important usage scenario besides providing a comprehensive overview of temporal evolution is that this representation can be used in combination with conventional renderings of field data to directly browse time steps with interesting shapes, abrupt changes or collisions. In addition to indicating shapes via luminosity, colour further depicts either time or classes of shapes. Here, unsupervised learning on the basis of the extracted shapes is employed to achieve shape-based colour mapping. Significant temporal overlap can be induced by some complex data sets that make it hard to depict a meaningful overview with one view. Although this cannot be avoided completely in general, our approach introduces different built-in measures to address and mitigate this issue. Our implementation further allows for the quick and progressive generation of our representation in parallel.

In the following, after reviewing related work in Section 2, we present our contributions of this work (cf. Figure 2 for an overview).

- Our temporally dense representation of shape evolution (Section 3):
  - … the convolution of a noise texture along contours depicts shapes via luminosity (Section 3.1),
  - … adaptive shape colour maps emphasize variations (Section 3.2),
  - … and temporal compositing enhances clarity (Section 3.3).
- Interactive temporal navigation and exploration for efficiently selecting time steps based on our dense representation (Section 4).
- A massively parallel and progressive implementation, and an evaluation with 2D and 3D data sets from different domains (Section 5).



(a) ***5Jets:*** *simulation a wave moving upward (time→color)* **(b)** ***Bottle:*** *Femto photography capturing a moving laser pulse [VWJ*13] (shape→color)*

Figure 1: *Temporal exploration with a pointer (circle) using our dense image-space representation, exemplified via a wave induced by five jets moving from top to bottom (a) and a laser pulse travelling through a bottle from left to right (b). Luminosity indicates contours while colour maps to time (a) or shapes (b). A rendering of the time step selected via pointer is shown as dark grey overlay (contour front coloured in pink).*

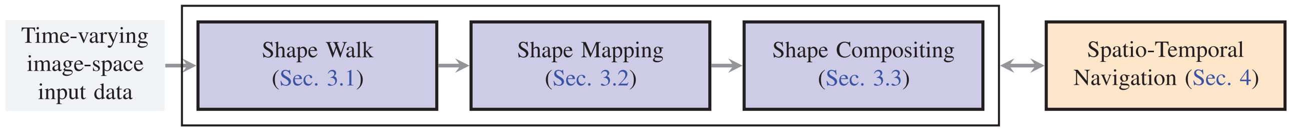

Time-varying image-space input data
Shape Walk (Sec. 3.1)
Shape Mapping (Sec. 3.2)
Shape Compositing (Sec. 3.3)
Spatio-Temporal Navigation (Sec. 4)

Figure 2: *Overview on the different components of our approach. First, we walk along shape contours individually for each cell (consisting of four adjacent pixels) of our image-space input data, convolving a noise texture to generate a luminosity-based shape map (*Shape Walk*, Section 3.1). Second, colour is either mapped to time or to classified shapes with an automatically generated table (*Shape Mapping*, Section 3.2).* Shape Walk *and* Shape Mapping *operate on individual time steps. Finally, the results are composited over time to yield our shape representation, also taking different means to address spatio-temporal overlap (*Shape Compositing*, Section 3.3). The representation provides the basis for interactive temporal exploration (*Spatio-Temporal Navigation*, Section 4). The whole process supports progressive computation (Section 5.6).*

We finally conclude our work in Section 6.

2. Related Work

Time-varying field data visualization. A variety of different approaches has been proposed for the visualization of time-dependent scientific 3D data. One line of work treats the data as a space-time hypercube, and applies extended classic visualization operations like slicing and projection [WWS03] or temporal transfer functions [BVMG08] (cf. Bach *et al.* [BDA*16] for an overview of techniques in this area). Alternatively, 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. Frey and Ertl [FE16] presented an approach to adaptively choose 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. To support such time step selection techniques with an expressive and fast-to-compute distance measure, they further introduced a technique to generate transformations between arbitrary volumes [FE17b, FE17a]. In contrast to these techniques, we do not require temporal selection and avoid the inherent hazard of missing crucial time steps or transitions. In another work, a representation has been presented that visualizes bounds of spatio-temporal processes to indicate where and when non-continuous changes occur [Fre18]. Although this approach is also temporally dense, shape information of involved objects is almost completely neglected. A comparison against these alternatives by example is provided in Section 5.5.

Another general approach to time-dependent volume visualization is based on feature extraction. Lee *et al.* [LS09a] extract trend relationships across variables in multi-field data. The concept of Time Activity Curves that contain each voxel's time series has been used as the basis for various techniques (e.g. [FMHC07, LS09b]). Wang *et al.* [WYM08] extract a feature histogram per volume block, characterize the local temporal behaviour and classify via k-means clustering. Isolating and tracking features has also been explored extensively (e.g. [SW97, JS06]). In this line of work, Widanagamaachchi *et al.* [WCBP12] propose a framework

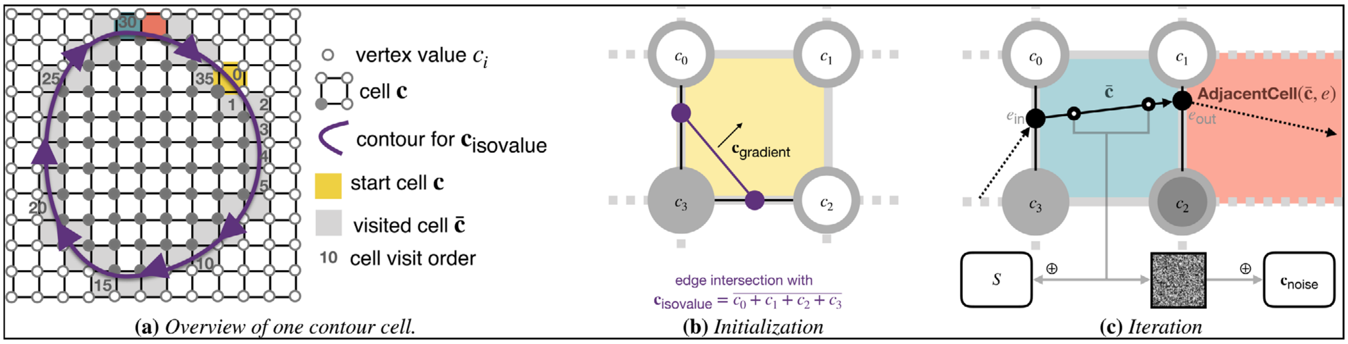

30
25
35
0
1
2
3
4
5
20
15
10
vertex value c_i
cell **c**
contour for $\mathbf{c}_{\text{isovalue}}$
start cell **c**
visited cell $\bar{\mathbf{c}}$
10 cell visit order
c_0
c_1
c_2
c_3
$\mathbf{c}_{\text{gradient}}$
edge intersection with
$\mathbf{c}_{\text{isovalue}} = \overline{c_0 + c_1 + c_2 + c_3}$
$\bar{\mathbf{c}}$
e_{in}
e_{out}
AdjacentCell($\bar{\mathbf{c}}$, e)
S
$\oplus$
$\mathbf{c}_{\text{noise}}$
(a) Overview of one contour cell.
(b) Initialization
(c) Iteration

Figure 3: *Illustration of Shape Walk (cf. Section 3.1). (a) For each cell* **c** *(yellow), we walk along an isoline (purple) passing through it, visiting cells* $\bar{\mathbf{c}}$ *on the way. (b) The isoline is defined by the average of the corresponding cell values. (c) When visiting the cells along the isoline, we convolve a noise texture to eventually yield a luminance value* $\mathbf{c}_{noise}$*, and collect samples S along the line as the basis for generating shape descriptors (cf. Figure 4a). We determine the next cell to visit by identifying which outgoing edge* e_{out} *to pass — considering that the current cell* $\bar{\mathbf{c}}$ *has been entered through* e_{in} *— and accordingly select the adjacent cell* $\neq \bar{\mathbf{c}}$ *(i.e.* $\bar{\mathbf{c}} \leftarrow$ **AdjacentCell**$(\bar{\mathbf{c}}, e_{in})$*).*

for the interactive exploration of dynamically constructed feature tracking graphs. Scale-space based methods and topological techniques have also been used in this context (e.g. [WDC*07, NTN15]). Schneider *et al*. [SWC*08] compare scalar fields on the basis of the largest shapes. Post *et al*. [PVH*03] and McLoughlin *et al*. [MLP*10] provide overviews of respective techniques for flow visualization.

Illustration-based techniques for time-varying data. Brambilla *et al*. [BCP*12] give an overview on illustrative techniques for flow visualization. In their classification scheme, the technique proposed in this work falls in the low-level category as we directly depict structures. High-level techniques implicitly consider semantics instead (e.g. cutaways or close-ups). In contrast to our technique, the vast majority of illustration-based visualization techniques for time-varying data are based on higher level data analysis (like tracking), increasing descriptiveness but requiring the specification of features. For instance, Joshi and Rheingans [JR05] apply different illustration techniques to time-varying data sets, including speed lines, flow ribbons and strobe silhouettes. Lu and Shen [LS08] generate interactive storyboards composed of individual volume renderings, enriched with descriptive geometric primitives. Meyer-Spradow *et al*. [MSRVH06] extract motion dynamics via block matching, and visualize the result via glyphs and speed lines. Eden *et al*. [EBG*07] employ cartoon style for rendering animations. Application-specific approaches have also been proposed, for example, showing the evolution of hurricanes [JCRS09] or ocean eddies [LSB*17].

Video Visualization. Our approach works in image-space and considers time-dependent data, and with this exhibits some relation to video visualization. According to the state-of-the-art report of Borgo *et al*. [BCD*], our proposed technique is low level (i.e. it operates on the pixel level), automatic and presents a single composite image providing background or context. In contrast, the majority of covered papers strongly rely on (high-level) object segmentation and tracking, essentially requiring the specification of features (similar to illustration-based techniques, cf. discussion above). Typical goals in classic video visualization include enabling of smart fast-forward capabilities (e.g. cf. Höferlin *et al*. [HKH*12]) and generating of (static) summaries (e.g. an overview on moving actors [CM10]). Most closely related to our approach are direct object manipulation techniques (e.g. [KDG*07, GKV*07, DRB*08, GGC*08, KWLB08, NNL14]). These methods extract objects and enable users to select them in the video and drag them along their respective motion trajectories. Akin to our approach, this allows for an image-space navigation in time. However, in contrast, we achieve this based on (low-level) shape contours without requiring explicit object or feature extraction.

3. Shape-based Spatio-Temporal Visualization

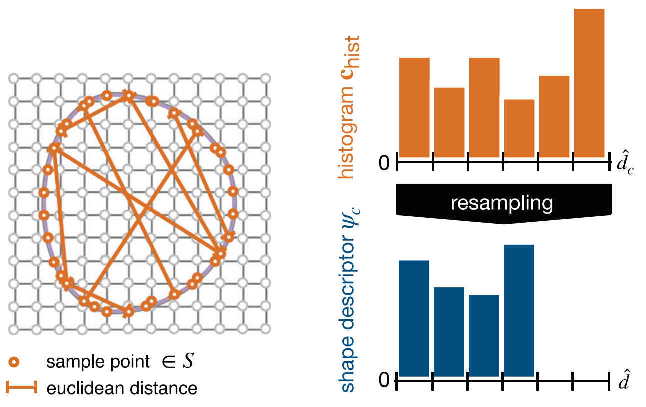
The typical approach for selecting time steps is to interact with a time slider. In this work, we aim to investigate an alternative for scientific visualization: based on a dense representation of the whole time series, we enable the selection of time steps directly in image space. Such a dense representation can also be useful for providing on overview on temporal processes. To achieve temporal density, some kind of (visual) abstraction needs to be performed to prevent significant overlap. Our approach does not require the extraction of complex features — in contrast to typical illustration-based and video analysis approaches — but employs image-space contours as low-level features. This enables a direct, pointer-based selection of time steps that depict, for instance, interesting shapes, abrupt changes of shapes or collisions. As a prerequisite for determining contours in image space, (3D) data sets are rendered using standard volumetric raycasting for user-defined camera positions and transfer functions. This then produces the time-varying image-space input data, which provides the basis to create our dense representation. Below, we discuss the individual steps of the approach to accomplish this.

3.1. Shape Walk

We first generate a luminance map depicting shapes in aggregated form. For this, we consider four adjacent pixels of an input image as cells **c** (pixels in the input images are also denoted as vertices of the cell in the discussion below, cf. Figure 3a). For each cell **c**, we

Algorithm 1. Shape Walk (Section 3.1) follows the contours passing through each cell, convolving a noise texture on the way. This determines the luminance value $\mathbf{c}_{\text{noise}}$ in our dense representation. Contour strength $\mathbf{c}_\omega$ and gradient $\mathbf{c}_{\text{gradient}}$ are used later in Shape Compositing (Section 3.3) for combining results over different time steps and enhancing contrast, respectively. The histogram of pairwise distances $\mathbf{c}_{\text{hist}}$ between samples $\boldsymbol{S}$ collected along the contour is later used to create a shape descriptor in Shape Mapping (Section 3.2)

```
 1: function SHAPE WALK($t \in T, \mathbf{c} \in \mathbf{C}^t$)
 2:   $\mathbf{c}_\omega \leftarrow \max(c_{0\dots3} \in \mathbf{c}) - \min(c_{0\dots3} \in \mathbf{c})$
 3:   if $\mathbf{c}_\omega = 0 \vee \max(\Delta_t c_{0\dots3} \in \mathbf{c}) \leq 0$
 4:     return                         ▷ skip cells with homogeneous or constant values
 5:   $\mathbf{c}_{\text{isovalue}} \leftarrow \overline{\sum c_{0\dots3} \in \mathbf{c}}$          ▷ average cell value (Fig. 3b)
 6:   $\bar{\mathbf{c}} \leftarrow \mathbf{c}$                    ▷ start contour walk with considered cell c
 7:   $e \leftarrow$ null              ▷ incoming intersection edge e is undefined initially
 8:   repeat                                        ▷ contour walk loop (Fig. 3a)
 9:     $\bar{\mathbf{c}}, e \leftarrow \text{AdjacentCell}(\bar{\mathbf{c}}, e)$              ▷ go to next cell (Fig. 3c)
10:     $\mathbf{c}_{\text{noise}} \leftarrow_+ \text{NoiseLookup}(\bar{\mathbf{c}})$       ▷ update noise value (Fig. 3c)
11:     $S \leftarrow_+ \text{ContourSample}(\bar{\mathbf{c}})$          ▷ collect samples (Fig. 3c)
12:   until $\mathbf{c} = \bar{\mathbf{c}}$             ▷ stop when cell of origin is reached again
13:   $\mathbf{c}_{\text{hist}} \leftarrow \text{PairwiseDistances}(S)$          ▷ →Sec. 3.2(a), Fig. 4
14:   $\mathbf{c}_{\text{gradient}} \leftarrow \text{ContourGradient}(E, e)$
15:   return c
```



(a) *pairwise distances* $\in S$ **(b)** *histogram: distance distribution*

Figure 4: *Generation of a histogram-based descriptor for shapes. (a) First, a set of samples S is collected for cells* **c** *during Shape Walk. With these, we compute the D2 shape descriptor by storing the distribution of distances between sample pairs in a histogram. (b) The histogram (orange) depicts distances in the range* $[0, \hat{d}_c]$*, with* $\hat{d}_c$ *being the largest occurring distance per shape. To make them directly comparable, we then resample the histograms with respect to* $\hat{d}$ *(the largest distance occurring across all histograms (blue)).*

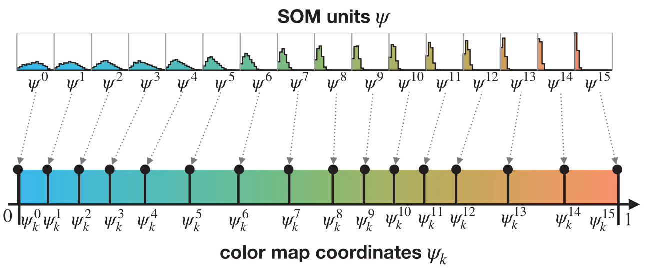

SOM units ψ
ψ^0 ψ^1 ψ^2 ψ^3 ψ^4 ψ^5 ψ^6 ψ^7 ψ^8 ψ^9 ψ^{10} ψ^{11} ψ^{12} ψ^{13} ψ^{14} ψ^{15}
0 ψ_k^0 ψ_k^1 ψ_k^2 ψ_k^3 ψ_k^4 ψ_k^5 ψ_k^6 ψ_k^7 ψ_k^8 ψ_k^9 ψ_k^{10} ψ_k^{11} ψ_k^{12} ψ_k^{13} ψ_k^{14} ψ_k^{15} 1
color map coordinates ψ_k

Figure 5: *For shape colour mapping, a (normalized) 1D colour map coordinate is assigned to each self-organizing map (SOM) unit (example given for the 5Jets data set). Here, the main idea is that the distance between shape descriptors of adjacent units* ψ^{i-1} *and* ψ^i *is relative to their distance in colour map coordinates* ψ_k^{i-1} *and* ψ_k^i*.*

compute one luminance value that results from convolving a noise texture along an isoline passing through the cell. In addition, further information required for later analysis is collected on the way. This procedure is discussed in detail below, supported by illustrations in Figure 3 and an outline of the procedure as pseudo-code in Algorithm 1.

We run Shape Walk for each cell $\mathbf{c} \in \mathbf{C}^t$ and each time step $t \in T$ of our input images. First, we assess the strength of a contour through $\mathbf{c}$ via the difference $\mathbf{c}_\omega$ in its four associated node values c_0, c_1, c_2 and c_3 (Algorithm 1, Line 2). This will later determine the visual impact of the cell when aggregating over time (Section 3.3). Homogeneous cells (i.e. $\mathbf{c}_\omega = 0$) are skipped (Lines 3 and 4). Also, to reduce temporal overlap when depicting evolving shapes, we only consider cells $\mathbf{c}$ featuring at least one vertex that has increased in value with respect to the previous time step (i.e. $\max(\Delta_t c_{0\dots3} \in \mathbf{c}) > 0$). This is a simple heuristic to identify the parts of a shape that are oriented towards the movement direction, under the assumption that the background exhibits a lower value than a moving object (this holds true for the examples given in this paper, in other cases adaptations or more elaborate heuristics can be required). The contour that we follow from cell $\mathbf{c}$ is defined by the isoline with the average value $\mathbf{c}_{\text{isovalue}}$ across all four vertex values $c_{0\dots3}$ (Line 5, Figure 3b).

Then, starting from cell $\mathbf{c}$ (Line 6), this isoline is followed until $\mathbf{c}$ is reached again (Lines 8–12, Figure 3a). For this, we exploit that isolines are closed by definition. To identify the next visited cell in every iteration, we identify the edge intersections with the isoline akin to the Marching Squares procedure (Line 8, Figure 3c). By definition, there are either two or four intersections. We determine the outgoing edge by identifying which intersected edge is topologically connected to the incoming edge e. We then accordingly proceed to the neighbouring cell with which this outgoing edge is shared. In the first iteration, there is no incoming edge (i.e. $e =$ null), and the edge intersection to start the walk with is selected randomly.

For each cell $\bar{\mathbf{c}}$ that we visit along the way, we look up a value from our noise texture and update c_{noise} via convolution (akin to LIC, Line 10, Figure 3c). We weight the impact of this value by the length of the shape line within the cell. We also collect a set S of 64 random samples from the incoming stream of points along the contour via reservoir sampling (Line 11). With this, the histogram $\mathbf{c}_{\text{hist}}$ of pairwise distances between samples $\in S$ is computed (Line 13). $\mathbf{c}_{\text{hist}}$ later serves as a basis to compute shape descriptors for Shape Mapping (cf. Figure 4, Section 3.2). Finally, we compute the gradient $\mathbf{c}_{\text{gradient}}$ by considering the isoline with $\mathbf{c}$ (Line 14, Figure 3b). $\mathbf{c}_{\text{gradient}}$ is used when compositing the final representation (cf. Section 3.3).

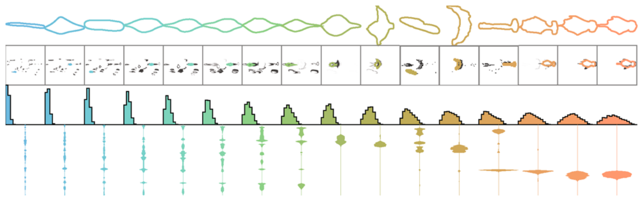
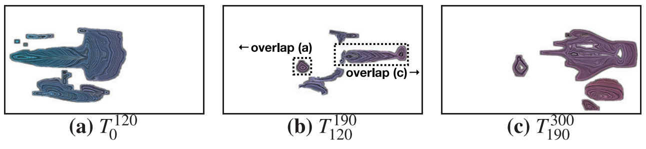


Figure 6: *SOM shape visualization at the example of the Bottle (Figure 1b). In the top row, from left to right, a representative shape for each unit is shown. In the row below, we indicate where classes of shapes occur in image space, and further down we depict the respective shape descriptor. In the bottom-most row, it is plotted how often a category of shapes appears over time along a vertical temporal axes (horizontal swings depict frequency at a certain point in time). Units are coloured according to the shape colour map.*



(a) T_0^{120} (b) T_{120}^{190} (c) T_{190}^{300}

Figure 7: *Temporal split example of our overview representation at the example of the Bottle data set. Here, the split into three parts mitigates temporal overlap stemming from the laser pulse covering the same area within and around the bottle in different stages of its evolution (cf. annotations in (b), discussion in Section 5.4, and representation covering whole time series in Figure 9a).*

3.2. Shape Mapping

We now integrate shape information in-place in the dense shape representation via colour and provide an additional overview on occurring shapes (e.g. cf. Figure 1b). For this, we generate a shape representation ψ_c from the sets of samples S collected during Shape Walk (a), and then use ψ_c to map shapes to colour and provide a shape overview representation (b).

(a) Shape Descriptor. We use the D2 shape descriptor to serve as a basis for quantifying the similarity between shapes. This choice is based on a study of Osada *et al*. [OFCD02] which found that this descriptor can be evaluated quickly and results in the best object classification results among five tested candidates, yielding distinctive and stable results in the context of translations, rotations, scales, mirroring, degeneracies, etc. The D2 shape descriptor requires a set of samples S from the contour, which we collect for each cell and time step during Shape Walk (Figure 4a). With these samples, we then compute the distribution of Euclidean distances $d(\cdot,\cdot)$ between all $\binom{|S|}{2}$ point pairs in S and store them as a histogram $\mathbf{c}_{\text{hist}}$ with 16 bins in our implementation (Figure 4a). In case there are less than $|S| = 64$ points in S, we continue to extend our signature with random pairs of points until $\binom{|S|}{2}$ combinations are considered. This is important to yield histograms with the same total number of entries. The width of histogram bins is individually determined via the largest occurring distance $\hat{d}_c = \max\{d(s_a,s_b)|s_a,s_b \in S\}$ in the respective shape. This means that the (uniform) width of the bins is chosen such that the last bin only just contains the largest determined distance $\hat{d}_c$. We carry out this procedure right after S has been determined during Shape Walk (Algorithm 1, Line 13). This means that we do not actually have to store S, but instead can save the significantly smaller histogram representation $\mathbf{c}_{\text{hist}}$.

We further need to make the different $\mathbf{c}_{\text{hist}}$ directly comparable. For this, we resample the histogram representations (which have been stored with respect to to the local maximum distance $\hat{d}_c$) to a shape descriptor ψ_c with 16 bins and the maximum distance $\hat{d} = \max\{\hat{d}_c | c \in \mathbf{C}^t\}$ across all shapes as a reference (b). This is done by splatting the value of each bin of $\mathbf{c}_{\text{hist}}$ to ψ_c using a Gaussian kernel. Although first determining $\hat{d}$ and then computing the histograms would avoid histogram resampling, this crucially would require a second pass of the Shape Walk procedure. According to our experiments, the differences due to resampling are negligible. We now have a set of shape descriptors $\psi_c \in \Upsilon$ (one for each shape). With this, shapes can easily be compared by via the Euclidean distances between their descriptors.

(b) Shape Colour Map and Visualization. Using the set of all shape descriptors $\psi_c \in \Upsilon$, we create (1) an overview representation of encountered shapes as well as (2) a colour map that assigns similar shapes to similar colours in our dense shape representation. Essentially, this encompasses two problems: (1) the projection of the shape descriptor into colour space, and (2) the clustering of shape descriptors. To address them, projection-based techniques (like t-SNE or UMAP) could be used to map shapes into the colour space (but the identification of clusters would need an extra step), or clustering techniques could be employed (but a meaningful ordering between clusters is required to yield a 1D colour map, cf. Figure 5). Instead, we employ (1D) SOMs [Koh98], which inherently features several useful properties for our application context: a neighbourhood function preserves similarity relationships across clusters (or units), the mapping to a 1D colour map is straightforward and representations of the respective clusters (units) are computed as an integral part of the technique.

In our SOM with a 1D topology of units $\psi^i \in \Upsilon_{\text{SOM}}$, each ψ^i is a weighted sum of the shape descriptors $\psi_c \in \Upsilon$ computed for each cell (cf. Figure 5 *(top)*). We follow a typical procedure for training SOMs. We first initialize each unit $\psi^i \in \Upsilon_{\text{SOM}}$ with a random shape descriptor $\psi_c \in \Upsilon$. In each of several training passes, we then loop over all shapes $\psi_c \in \Upsilon$ in random order, and for each identify the unit ψ^i it is closest to. We then update the histogram representations of ψ^i and the units in its vicinity by pushing them towards ψ_c. The considered neighbourhood decreases with an increasing number of passes, as does the impact on the respective histograms. Similarly, the impact of the update decreases the further the distance to the closest unit is. We use 16 units ($\psi^0 \ldots \psi^{15}$), and run 256 training passes with an initial neighbourhood radius of eight as well as a unit update rate of 5%, decreasing both by $\times 0.98$ after each pass. The impact of different numbers of units is evaluated in Section 5.3.

Finally, we map the resulting units $\psi^i \in \Upsilon_{\text{SOM}}$ to colour (cf. Figure 5). For this, we assign (normalized) 1D colour map coordinates ψ_k^{i} to each unit ψ^i such that the distance between shape descriptors of adjacent units corresponds to distances in colour map coordinates ψ_k^{i}:

$$\psi_k^{\text{i}} = \frac{\sum_{j=1}^{i} \Delta(\psi^{j-1},\psi^j)}{\sum_{j=1}^{15} \Delta(\psi^{j-1},\psi^j)}. \quad (1)$$

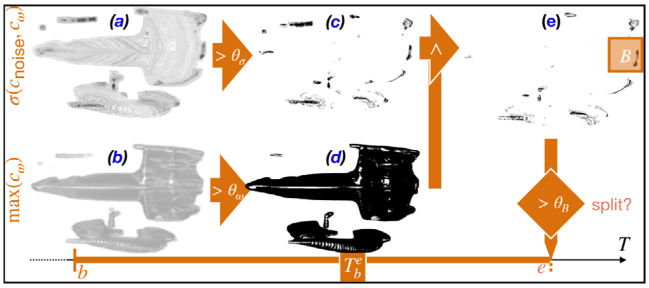


Figure 8: *Illustration of split at the example of the Bottle (refers to first split at T_0^{120} Figure 7a). (a) First, we assess the weighted standard deviation of noise values $\sigma(\mathbf{c}_{noise}, \mathbf{c}_\omega)$ and (b) the contour strength* $\max(\mathbf{c}_\omega)$*. (c) We then create masks depicting cells with (d) significant overlap ($> \theta_\sigma$) and (e) strength ($> \max(\mathbf{c}_\omega)$). (e) Their combined mask is then checked for a region B containing numerous active cells ($> \theta_B$). If this is the case, the time series is split at e.*

With this, we obtain the colour of shape ψ_c by identifying its closest unit $\psi^i{}_i$ and accessing the colour map via ψ_k^{i}.

Besides colour mapping, shape information is also presented in a dedicated view (Figure 6). In the top row, for each unit ψ^i, we depict the shape ψ_c with the smallest distance to ψ^i as representative. In the row below, we indicate where these shapes occur in image space, and further down we show the respective shape descriptor (in the form of a histogram). In the bottom-most row, we illustrate the changing frequency of occurring shapes over time.

3.3. Shape Compositing

We now composite the final image. (a) Optionally, we first partition the time series to reduce temporal overlap in our combined dense representation. In our representation, we use two channels to convey information: (b) colour (hue) to depict time or classes of shapes and (c) texture (luminosity) to represent shape contours.

(a) Detecting Shape Overdraw and Split. Our approach aims to generate a comprehensive overview of shape evolution across a time series. Especially for complex processes, there can be substantial overdraw in image space which can lead to loss of detail and visual clutter. To address this, we provide the option to adaptively split our representation into individual representations of time subsequences whenever significant overdraw is detected. For example, in Figure 7, the split prevents temporal overlap stemming from the laser pulse covering the same area within and around the bottle in different stages of its evolution (cf. Section 5.4 for a detailed discussion).

We check for every time window T_b^e ($t \in [b, e]$ with incrementally increasing e) whether there is an area in image space with substantial overlap, and then split accordingly (Figure 8). First, we assess the variation of luminance values $\mathbf{c}_{\text{noise}}$ of each cell over time to quantify overdraw. For this, we use the weighted standard deviation $\sigma(\mathbf{c}_{\text{noise}}, \mathbf{c}_\omega)$ over the time window $t \in T_b^e$ with weights $\mathbf{c}_\omega$:

$$\sigma(\mathbf{c}_{\text{noise}}, \mathbf{c}_\omega) = \sqrt{\frac{\sum_{t \in T_b^e} \mathbf{c}^t{}_\omega (\mathbf{c}^t{}_{\text{noise}} - \overline{\mathbf{c}_{\text{noise}}})^2}{\frac{M-1}{M} \sum_{t \in T_b^e} \mathbf{c}^t{}_\omega}}. \qquad (2)$$

Here, $\overline{\mathbf{c}_{\text{noise}}} = \frac{\sum_{t \in T_b^e} \mathbf{c}^t{}_\omega \mathbf{c}^t{}_{\text{noise}}}{\sum_{t \in T_b^e} \mathbf{c}^t{}_\omega}$ represents the weighted mean and $M = \sum_{t \in T_b^e} [\mathbf{c}^t{}_\omega > 0]$ denotes the number of non-zero weights. The standard deviation $\sigma(\mathbf{c}_{\text{noise}}, \mathbf{c}_\omega)$ is then used as a main part of our criterion to identify temporal overlap in the time window T_b^e (Figure 8 (a)). We further consider the maximum strength $\max(\mathbf{c}_\omega)$ occurring over time with the intention to only consider overdraw produced by pronounced contours (Figure 8 (b)). Finally, we split if there exists an image region of size B featuring at least θ_B cells with significant variation in luminance (exceeding threshold θ_σ, Figure 8 (c)) and pronounced contours (i.e. $\mathbf{c}_\omega$ is larger than threshold θ_ω, Figure 8 (d)):

$$\exists B (\sum_{\mathbf{c} \in B} [\sigma(\mathbf{c}_{\text{noise}}, \mathbf{c}_\omega) > \theta_\sigma \wedge \max(\mathbf{c}_\omega) > \theta_\omega] > \theta_B). \qquad (3)$$

We used parameter settings $\theta_\sigma = 0.02, \theta_\omega = 0.5, B = 64^2$ and $\theta_B = 0.1B$. The computation of $\sigma(\mathbf{c}_{\text{noise}}, \mathbf{c}_\omega)$ and $\max(\mathbf{c}_\omega)$ can be computed iteratively at interactive rates, they only need to be updated when incrementing e. Likewise, Equation (3) can be implemented efficiently using summed-area tables (cf. timings in Section 5). Note that Shape Mapping is always computed across the whole time range T, and is therefore not affected by the splitting.

(b) Colour (Hue). Time or shape can be encoded via colour. For both, we use isoluminant colour maps, that is, the luminosity is constant across all members of the map [Kov15]. This allows to separate clearly between hue and luminance. For time, we use a colour map directly while for shape we employ the result from Section 3.2 (Figure 5). With both encodings, colour is assigned to every cell and time step through which a contour passes (i.e. whenever the skip criterion in Algorithm 1, Line 3 is not fulfilled).

We compute the aggregate colour of each cell $\mathbf{c}_{\text{colour}}$ as follows. First of all, for each cell and time step t with non-zero weight $\mathbf{c}^t{}_\omega > 0$, we determine the normalized colour map coordinates k^t, depending on the colour mode:

$$k^t = \left\{ \begin{array}{ll} {}^t/_{|T|}, & \textbf{if } \text{time} \rightarrow \text{colour} \\ \psi_k^{\text{i}} \text{ with } \psi_c \in \psi^i, & \textbf{if } \text{shape} \rightarrow \text{colour (cf.Figure5)} \end{array} \right) . \qquad (4)$$

For time, this is simply the respective normalized point in time t. For shape colouring, we identify the closest SOM unit ψ^i to shape descriptor ψ_c of the respective cell, and use the corresponding coordinates ψ_k^{i} (cf. Section 3.2, Figure 5). For each cell, we compute the weighted standard deviation $\sigma(k, \mathbf{c}_\omega)$ indicating uncertainty, with values k^t and weights $\mathbf{c}^t{}_\omega$ over time $t \in T$ (akin to Equation (2)).

For every time step t at each cell, we convert the colour obtained from the map to the HSV colour space. We then adjust the colour saturation via the deviation of values $\sigma(k, \mathbf{c}_\omega)$. For this, we multiply the saturation channel by $\max(1. - \sigma(k, \mathbf{c}_\omega) \cdot m, 0)$, with m being a parameter indicating how fast the saturation should decrease (we use $m = 5$ throughout this paper). Next, the modified colour is converted back to RGB space, yielding $\mathbf{c}^t{}_{\text{colour}}$. Finally, we composite a weighted sum of the colours over time, resulting in the final $\mathbf{c}_{\text{colour}}$ for a cell:

$$\mathbf{c}_{\text{colour}} = \frac{\sum_{t \in T} (\mathbf{c}^t{}_\omega \mathbf{c}^t{}_{\text{colour}})}{\sum_{t \in T} \mathbf{c}^t{}_\omega}. \qquad (5)$$

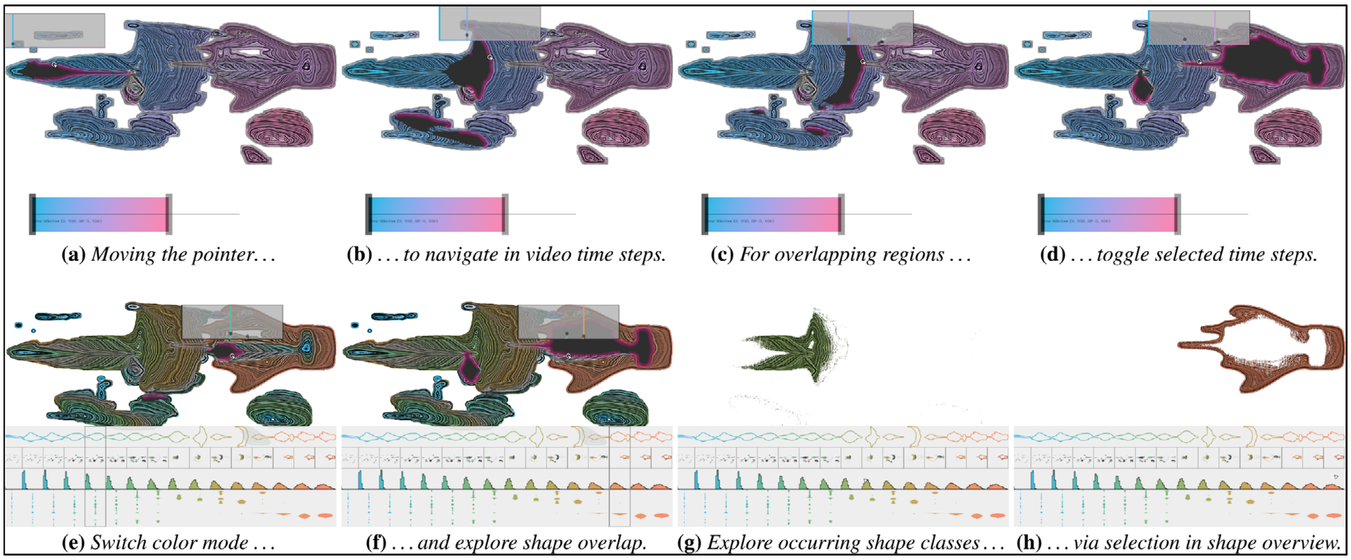

(a) Moving the pointer...
(b) ... to navigate in video time steps.
(c) For overlapping regions ...
(d) ... toggle selected time steps.
(e) Switch color mode ...
(f) ... and explore shape overlap.
(g) Explore occurring shape classes ...
(h) ... via selection in shape overview.

Figure 9: *Exploration sequence at the example a laser pulse travelling through a bottle captured via Femto photography [VWJ*13].*

(c) Luminosity. We use a three-step approach to yield clearly distinguishable shapes with high contrast between neighbouring cells. First of all, similar to Equation (5), we compute a weighted sum of the noise $\mathbf{c}_{\text{noise}}$ and the gradient magnitude $|\mathbf{c}_{\text{gradient}}|$. Second, we determine the difference in value $\Delta\mathbf{c}_{\text{noise}}$ in the direction of gradient $\mathbf{c}_{\text{gradient}}$ from cell position c: $\Delta\mathbf{c}_{\text{noise}} = \mathbf{c}_{\text{noise}} - (c + {}^{\mathbf{c}_{\text{gradient}}}/_{|\mathbf{c}_{\text{gradient}}|})_{\text{noise}}$. Directly using the noise value $\mathbf{c}_{\text{noise}}$ also yields decent results, but we found that the difference based on the gradient yields clearer, higher contrast images. Third, we compute luminosity $\mathbf{c}_{\text{luminosity}}$ by applying a sigmoid function to $\Delta\mathbf{c}_{\text{noise}}$ for further contrast enhancement:

$$\mathbf{c}_{\text{luminosity}} = \frac{1}{1+\exp(\Delta\mathbf{c}_{\text{noise}})}. \tag{6}$$

The final output for a cell is then simply determined via the multiplication of $\mathbf{c}_{\text{luminosity}}$ and $\mathbf{c}_{\text{colour}}$ from (b).

4. Interactive Temporal Navigation and Exploration

We now introduce modalities for interactive exploration based on our dense representation. The image-based selection of time steps directly allows for temporal navigation by hovering with a mouse pointer over the location of interest (e.g. Figure 9a–c). For a cell $\mathbf{c}$ selected by a pointer, we obtain the respective sequence of weights $\mathbf{c}^t{}_{\omega}$ over time t. Peaks in the sequence of $\mathbf{c}^t{}_{\omega}$-values indicate points in time with a distinct shape boundary. Accordingly, we extract maxima to capture when this occurs, smoothing the signal beforehand using a box filter for the sake of stability. We present this sequence of values $\mathbf{c}^t{}_{\omega}$ in a semi-transparent overlay (horizontal axis: time, vertical axis: weights, cf. Figure 9a–f). Depending on the mode, the shown line is coloured either with respect to time or shape. Maxima are indicated via filled circles, and a vertical line depicts the currently selected point in time.

Several maxima indicate spatio-temporal overlap, that is, a cell is covered by distinct shapes at different points in time. During exploration, we automatically choose the maximum whose respective time step is closest to the current one. A user may iterate through different selections of maxima by pressing a button (e.g. Figure 9c and d, and Figure 9e and f). Finally, we render the selected time step (dark grey), and composite the selected time step in-place over our dense contour representation. As discussed above in Section 3.1, only contour parts with increasing cells are considered (indicated in pink).

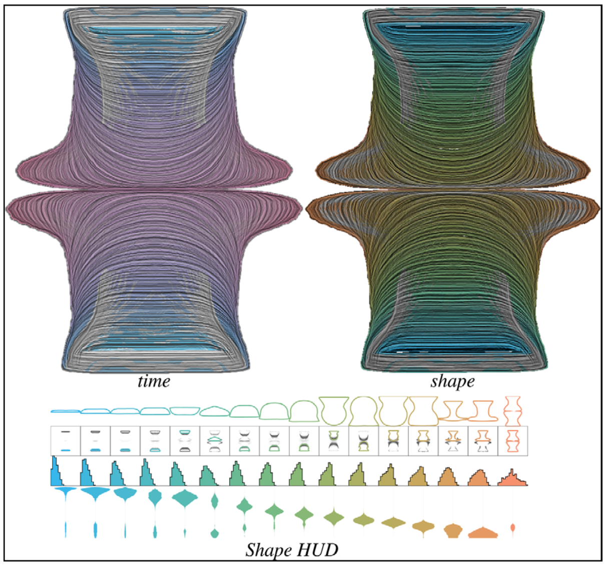


Figure 10: *A cold plate on top and a hot plate on the bottom induce colliding flow of cold and hot air.*

In the bottom, we show information depending on the current colour mapping mode (cf. Section 3.3), that is, either with respect to time (Figure 9a–d) or shape (Figure 9e–h). A user can interactively

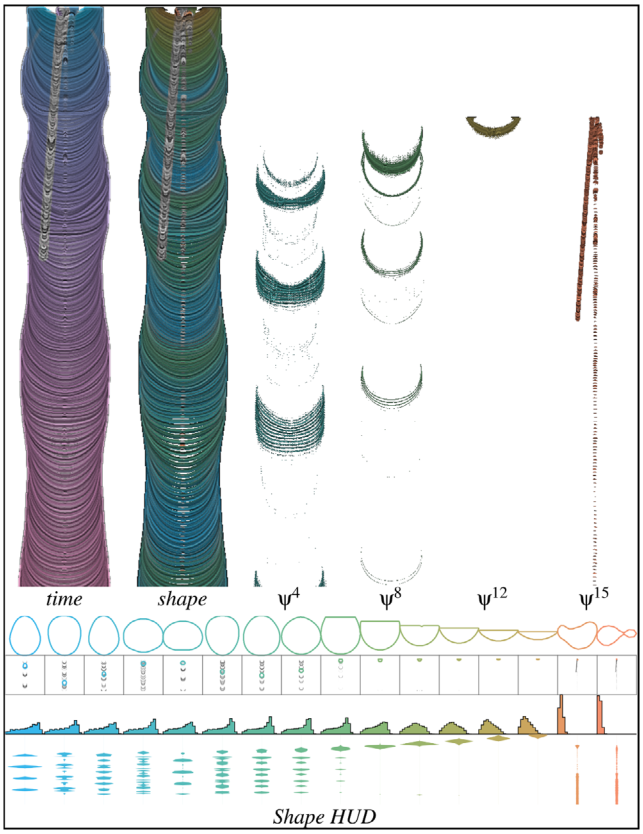


Figure 11: *Drop forming on top and falling down, with two smaller droplets following afterwards.*

switch between the two modes. For time, we depict the time transfer function that can further be used as an interface to only select a temporal subrange of the video (e.g. in Figure 9, the first two-thirds of the whole time series are considered). In the shape mode interface (introduced above in Section 3.2), we can filter for a specific category of shapes by selecting it directly via the pointer. Then only the shapes belonging to this category are shown, allowing a user to explore where (and when) different types of shapes occur (Figure 9g and h).

5. Results

We evaluate our approach with 2D and 3D time-dependent scalar fields (volume raycasting generates images in the case of 3D volumetric data, cf. Section 3). These are the used data sets:

- 5Jets *(resolution*: 128^3, *source: CFD simulation, Figure* 1a),
- Bottle (900×430, *Femto photography* [VWJ*13], *Figure* 1b),
- Hot Room (181×91^2, *CFD simulation, Figure* 10),
- Droplet Measurement (182×878, *experiment, Figure* 11),
- Droplet Simulation (256^3, *multiphase flow sim.* [EEG*16], *Figure* 12),
- von Kármán (101×301, *2D CFD simulation, Figure* 13),
- λ_2 (529^3, λ_2 *criterion for vortex extraction* [JH95], *Figure* 15).

The number of time steps and performance measurements of individual steps are provided in Table 1. We utilize an image resolution of 1024×1024 throughout this paper for the sake of comparability. Below, we first discuss the expressiveness of our dense shape representation (Section 5.1), before demonstrating the utility of our approach for interactive data exploration (Section 5.2). We then look at the impact of the number of SOM units (Section 5.3), evaluate the results of our temporal splitting approach (Section 5.4) and compare against other techniques creating spatio-temporal overviews Section 5.5. We then present the implementation of our progressive system and timings (Section 5.6), before finally discussing current limitations and directions for future work (Section 5.7).

5.1. Dense Shape Representation

First, we discuss the proficiency of our dense shape representation in providing an overview on spatio-temporal behaviour.

5Jets (Figure 1a). This data result from a simulation of five jets entering a region. Temporal colouring and texture depict the movement of the main wave going from the bottom to the top, with its extent decreasing in the process (the volume rendering in dark grey shows this wave). The lighter grey areas in our dense representation are due to temporal overlap. They indicate individual parts that split off the main wave, and travel onward with lower velocity.

Bottle (Figure 1b **and Figure** 9). The data set captures a laser pulse travelling through a bottle (obtained via Femto Photography [VWJ*13]). On its way from left to right, the laser gets reflected and scattered in different ways, inducing different phases. Early on, the pulse forms a structure with a sharp tip, as can be seen nicely from the texture of our visualization. Later, it smoothly transitions into a wave with a large vertical extent. Finally, a small laser pulse reappears that initially moves left-to-right and eventually fills the top of the bottle with a much larger shape. This change in shape is clearly indicated via shape colouring. In both colour modes, the grey areas indicate overlap. However, with shape colouring the colour has been pushed significantly more towards grey in comparison to time colouring (cf. top and bottom row in Figure 9, respectively). This is due to the fact that, as indicated above, the shape changes rather quickly in a small temporal window (i.e. lower variance) while the change in shape is quite significant in a short time frame (i.e. higher variance). Note that we also exemplify our temporal splitting approach by means of this data (cf. Figure 7, discussion below in Section 5.4).

Hot Room (Figure 10). The data set depicts colliding flow of cold and hot air, induced by a cold and hot plate on the top and the bottom, respectively. The representation shows that the width of the two volumes of air which develop initially approximately equals the area of the plates. The shapes slowly transform while moving towards each other (cold air going down, hot air moving upward). The shape HUD accordingly depicts both classes of shapes and when they occur. Once the two masses of air collide, their shape transforms significantly. It becomes thinner closer to their respective plates, but broader in the centre where they meet. Time colouring indicates the symmetric bottom-up and top-down movement of the respective air masses, whereas shape colouring emphasizes the smooth changes in shape (as also indicated by the shape

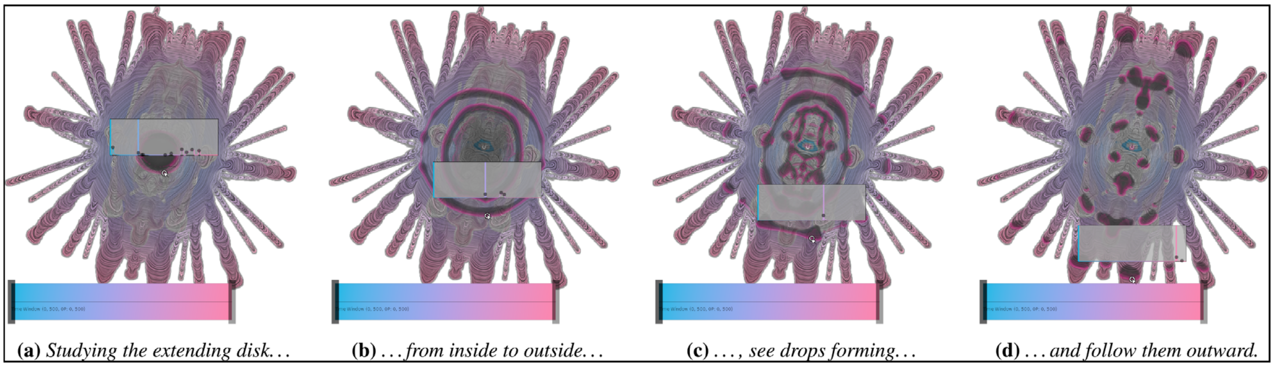

(a) Studying the extending disk…
(b) …from inside to outside…
(c) …, see drops forming…
(d) …and follow them outward.

Figure 12: *Two asymmetrically colliding drops form a disk that extends outward and eventually decomposes into individual droplets again.*

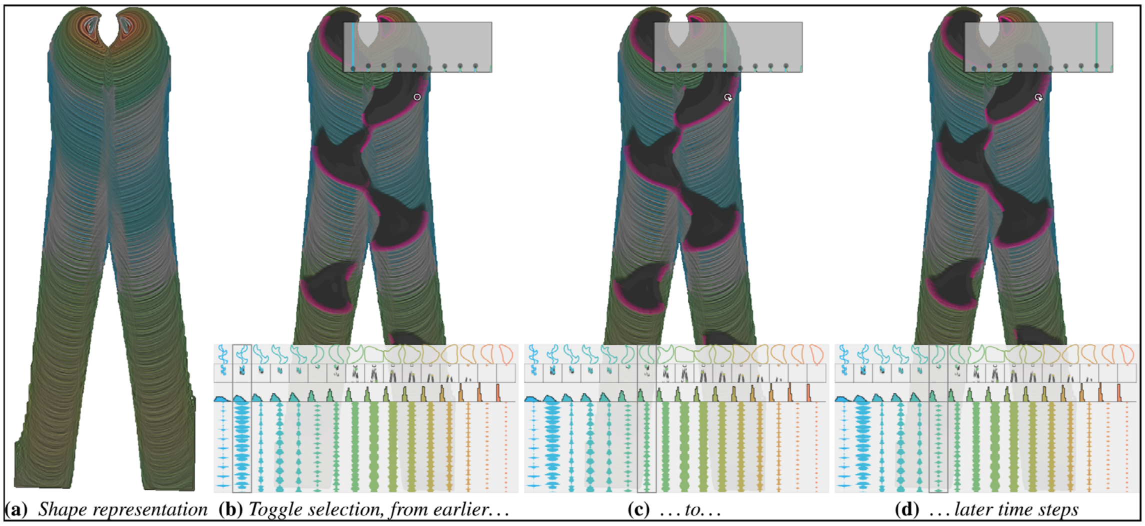


Figure 13: *2D CFD simulation of a von Kármán vortex street. Texture and colouring depict a smooth change of shapes from top to bottom overall. However, there are different shapes occurring in the same region of this recurrent process (grey areas). We investigate this by switching between shapes and find that in later time steps shapes are slightly different, for example, exhibiting earlier separation.*

visualization in the bottom). With both colour mappings, the grey areas depict overlap in similar areas, indicating that significant changes in shape and temporal overlap with a larger time distance coincide. Most prominently, this shows in the overlap between the broad shape that initially moves upward and the more narrow shape that develops after the collision.

Droplet Experiment (Figure 11). The data capture an experiment of a drop forming at the top, and eventually falling to the bottom. In the process, also two smaller droplets form and fall down later. In both time and shape colouring, these later droplets are indicated in grey as they produce temporal overlap and exhibit a significantly smaller shape. The representation also indicates that the large droplet changes shape in a recurrent fashion (changing colour from blue to green and back). This can be investigated in more detail when directly selecting individual shape classes belonging to different SOM units ψ^i(Figure 11, right). This separates different droplet phases ($i = 4$, $i = 8$, and $i = 12$), and also depicts the smaller droplets belonging to the same category ($i = 15$).

5.2. Exploration

We introduced the user interaction modalities of our approach in Section 4, and exemplified them by analyzing the path and transformation of the laser pulse in the Bottle data (Figure 9). Below, we provide additional examples of temporal exploration.

Droplet Simulation (Figure 12). The data set depicts a simulation of two droplets colliding asymmetrically. There are many different processes in the centre where the droplet collision occurs, and accordingly the data exhibit significant temporal overlap (as indicated by the grey areas in the representation). At the example of this data set, we primarily exemplify the temporal exploration with the pointer. First, the two initial droplets merge and form a disk in the centre (Figure 12a). We then follow the disk outward as it extends in size and increasingly separates into individual droplets (Figure 12b and c). We further track an individual droplet with the pointer as it moves outward along a finger-like structure (Figure 12d). With this, we can also study how other droplets progress simultaneously. This

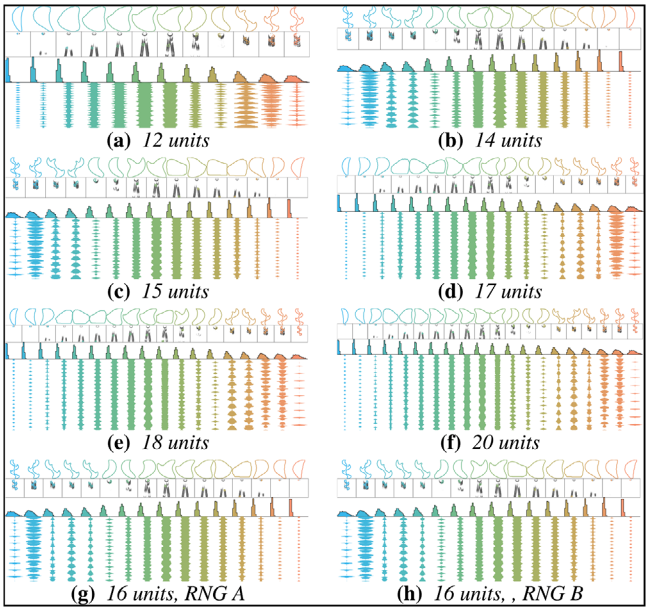


Figure 14: *(a–f) Study of the impact of different numbers of units and (g–h) random number initialization at the example of the von Kármán vortex street (cf. Figure 13). The depicted shapes and their frequency are stable across all numbers of units, showing only small differences due to different resolutions. The most prominent deviation is due to inverted left-to-right orderings.*

demonstrates the direct and precise temporal navigation by selecting processes of interest in image space (e.g. droplets moving outward). Despite temporal overlap, we smoothly progress in time starting from the currently selected time step (cf. accompanying video).

Beyond supporting navigation, our dense shape representation directly gives an overview on the amount of droplets ejected from the centre via the amount of finger-like structures. These structures also depict the trajectories of the droplets and indicate velocity. The finger widths further provide information on the size of the droplets. Furthermore, the texture in the larger fingers indicates droplet oscillations (e.g. [KBE*17]).

von Kármán (Figure 13). In this analysis of a CFD simulation of a flow around a cylinder, the focus of interest is on regions of high velocity magnitude. In the visualization, the cylinder is located at the top, and aforementioned regions move from the top to the bottom in two lanes. From the texture of the representation and the shape colouring in Figure 13(a), it can be seen that the shape of the high-velocity regions changes during the course of downward movement. The peaks shown in the mouse selection overlay (depicted via filled grey circles in Figure 13b–d) illustrate that shapes repeatedly pass through at constant frequency of approximately every 30 time steps. The fact that large parts of the representation appear in a shape colour (instead of grey) indicates that the respective shapes do not change significantly in different iterations of the process.

Nevertheless, there are also some grey areas depicting overlap of different shapes. To investigate this in more detail, we move the pointer to such a region and toggle between selected time steps (Figure 13b–d). First of all, while switching between respective time steps, we can see from the time step renderings depicted in dark grey that quite similar states are repeated over and over again. However, there are slight differences in shape which cause our approach to detect shape overlap. Essentially, there are smaller variations in the time series that lead to individual shapes separating earlier or later on their way down. Although in the first occurrence the shapes on the top are still connected (as indicated by the blue colour and the selection in the shape HUD, Figure 13b), they are further apart in later cases resulting in disconnected shapes (Figure 13c and d). Identifying and pointing out these small variations can be useful for analysis. In particular, this can point out small variations in recurrent behaviour that otherwise appears almost identical. In this case, further simulations would be required to exclude or confirm whether detected variations are a result of small differences in physical processes or due to insufficient spatio-temporal data resolution.

5.3. Number of Units for Colour Mapping

We now investigate the sensitivity of the results regarding the number of units of the SOM by means of the von Kármán data set. Figure 14 (a–f) shows that across all numbers of units similar shapes and descriptors have been identified (i.e. the results are stable, and largely independent of the exact number of chosen units). The sole significant difference are reversed orderings, which has practically no impact for the evaluation. In cases in which persistence is crucial, that is, when a user aims to explore colour maps created with different numbers of units, the orderings could automatically be reversed to account for this. Finally, the training procedure of the SOMs includes some randomness (e.g. in the initialization of the units, cf. Section 3.2 (b)). However, choosing different seeds only has negligible impact on the result. We empirically confirmed this for all number of units in this study and exemplify this here in Figure 14(g) and (h).

5.4. Temporal Split

We evaluate the results of our temporal splitting approach by means of two cases: the Bottle (Figure 7) and the λ_2 (Figure 15) data set. The main goal of this technique is to address temporal overdraw.

With the Bottle data set, the technique results in a representation temporally split into three partitions (Figure 7). This addresses overlap issues that can be seen in Figures 1(b) and 9. The first split (between Figure 7a and b) deals with overlap induced by a change of phases in the evolution of the laser pulse, most prominently a large reflection spot occurring slightly left and below of the centre. The second split (between Figure 7b and c) addresses overlap of the initial thin pulse travelling to the head of the bottle and the subsequent illumination of the whole upper part of the bottle by the pulse.

The λ_2 data set initially features a clear cross-like structure, that increasingly decomposes over time, resulting in complex and heavily overlapping small objects towards the end (Figure 15).

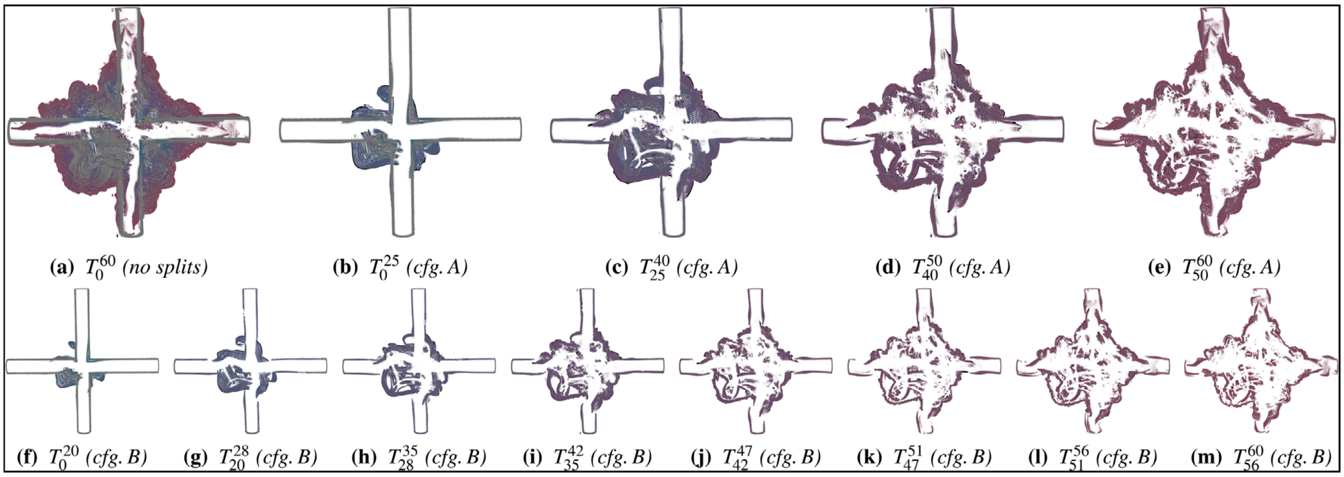

(a) T_0^{60} *(no splits)*
(b) T_0^{25} *(cfg. A)*
(c) T_{25}^{40} *(cfg. A)*
(d) T_{40}^{50} *(cfg. A)*
(e) T_{50}^{60} *(cfg. A)*
(f) T_0^{20} *(cfg. B)*
(g) T_{20}^{28} *(cfg. B)*
(h) T_{28}^{35} *(cfg. B)*
(i) T_{35}^{42} *(cfg. B)*
(j) T_{42}^{47} *(cfg. B)*
(k) T_{47}^{51} *(cfg. B)*
(l) T_{51}^{56} *(cfg. B)*
(m) T_{56}^{60} *(cfg. B)*

Figure 15: *Temporal splitting for the λ_2 data set featuring complex, fine-granular spatio-temporal processes. (a) Without splits, the representation of the complex processes exhibits significant temporal overlap. (b–e) Splitting addresses this issue (using the same parameter settings as for the Bottle in Figure 7,* (config A)*). The first phase of the time series in which the initial structure collapses exhibits comparably little overlap and the representation in (b) depicts a long sequence covering almost half the series. Afterwards the behaviour gets more chaotic and splits occur every 10 to 15 time steps. (f–m) A stricter parameter setting (reducing θ_ω from 0.5 to 0.3,* (cfg. B)*) yields finer splits with a higher frequency (with the first split at time step 20 (f), later subdividing every four to eight time steps (g–m)).*

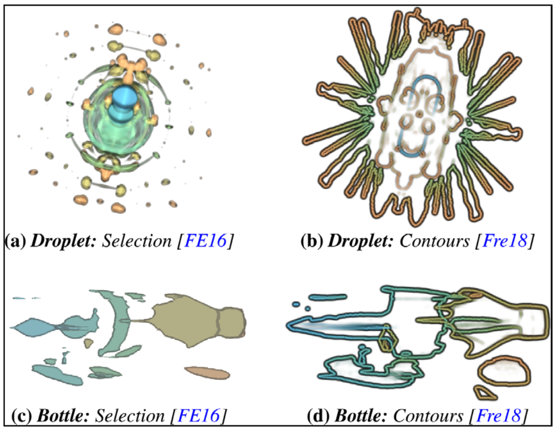

(a) ***Droplet:*** *Selection [FE16]*
(b) ***Droplet:*** *Contours [Fre18]*
(c) ***Bottle:*** *Selection [FE16]*
(d) ***Bottle:*** *Contours [Fre18]*

Figure 16: *Comparison to other techniques generating spatio-temporal overviews (selection and spatio-temporal contours) by means of the Droplet Simulation and the Bottle data set.*

Accordingly, while an overview representation of the whole time series allows to gain some insights into temporal progression, there are large areas of significant overlap, hindering the analysis of shape outlines (Figure 15a). Splitting the time series (with the same settings as for the Bottle discussed above) yields a temporal partitioning reflecting the temporal progression (b–e). The first representation depicts a longer temporal sequence that shows the initial decomposition and the curved shapes that are forming (b). The next splits occur after 15 time steps, and successively after 10, respectively, due to significant overlap (c–e). Decreasing θ_ω from 0.5 to 0.3 yields a qualitatively similar, yet finer decomposition with less overlap (f–m).

In general, while splitting helps to deal with temporal overlap, it comes at the cost of multiple representations, and making sense of processes with small, complex shapes and heavy overlap still remains challenging (cf. discussion in Section 5.7).

5.5. Comparison to Alternatives

We compare the results of our approach to other techniques creating spatio-temporal overviews, namely temporal selection [FE16] and spatio-temporal contours [Fre18] (cf. Section 2). Time is mapped to colour in both techniques. We use the Droplet Simulation and the Bottle data set with the same camera configuration and transfer function as in the rest the paper.

Temporal selection maintains the full spatial information for chosen time steps, and generally provides an intuitive overview (Figure 16a and c). However, only a small subset of time steps are presented. This means that interesting processes or events may be missed, and transitions between selected time steps are not represented. For instance, in the Droplet Simulation, the temporal changes in shape along the paths of the droplets splashing out and the decomposition of the disk are not represented (Figure 16a). The spatial coverage of processes also cannot be assessed to their full extent, for example, how far the disk and the droplets splash out. In the Bottle, different stages of the laser pulse in its evolution through the bottle are missing, for example, from the wide pulse depicted in green to the filled bottle head in light brown (Figure 16a). In contrast to this temporally sparse technique, our approach is able to present shape information in a temporally dense way. However, this requires a more abstract representation (based on shape outlines). Nevertheless, our approach also supports the analysis of spatial aspects in full detail via the direct exploration of renderings (as demonstrated in Figure 16c).

Table 1: *Performance measurements on GPU (CUDA) and CPU (OpenMP). Our Mapping implementation only runs on the CPU.*

Data set	Shape walk		Mapping	Rendering		#shapes
(Name, time steps)	(Section 3.1, in s)		(Section 3.2, in s)	(Section 3.3, in s)		(in thousands)
	GPU	CPU	CPU	GPU	CPU	
5Jets, 340	42.7	283.4	152.1	0.1	1.4	3114
Bottle, 300	8.2	132.2	86.2	0.1	1.2	1828
Drop. Meas., 315	3.4	40.2	14.7	0.1	1.3	294
Drop. Sim., 500	35.8	313.1	551.4	0.1	2.1	10 281
Hotroom, 50	9.1	127.9	57.0	0.0	0.2	1268
von Kármán, 300	64.5	239.9	302.2	0.1	1.3	5559
λ_2, 60	59.7	623.7	104.7	0.0	0.3	1891

Spatio-temporal contours largely neglect shape information, but present data in a temporally dense way (Figure 16b and d). For instance, contours indicate the paths of individual droplets (Figure 16b). They further outline boundaries in spatio-temporal processes, for example, the sharp tip formed by the pulse early on in blue, and in green the endpoint of the wave with a large vertical extent through the centre of the bottle (Figure 16d). However, this representation does not directly capture spatial structure, and neglects (non-abrupt) shape changes that occur smoothly and continuously. For example, in the Droplet Simulation data, the lanes going outward from the splash only represent droplet paths, but not the shape of individual droplets (Figure 16b).

In contrast to both techniques, our approach presented in this paper is able to depict shape information in a temporally dense fashion. It maintains both structural information (like the temporal selection technique) and depicts all time steps outlining the bounds of spatio-temporal processes (like the contour approach). For instance, for the Droplet Simulation, we present both the path of splashing out droplets as well as their changes in shape (Figure 12), and for the Bottle we show the outlines of the laser pulse and provide information regarding its spatial form (Figure 1b and Figure 9). In addition, our representation directly supports smooth image-based temporal browsing. Finally, spatio-temporal overlap is an issue not only for the approach presented in this paper, but also for both the selection-based and spatio-temporal contour technique. We address this issue via splitting (cf. Section 3.3 (a)). Potentially, the general approach to detect and prevent spatio-temporal overlap could also be implemented for the other techniques in an adjusted form in future work.

5.6. Timings and Progressive System Implementation

We evaluate the implementation of our approach on a machine equipped with an Intel Core i7-6700 CPU, 64 GB of memory, and two NVIDIA GPUs. We use a GeForce GTX 1070 (with 8 GB of memory) for the interactive display via OpenGL. For generating our dense representation, we utilize a TITAN X with 12 GB of memory. Each individual step for computing our shape-based spatio-temporal visualization was designed to efficiently utilize parallel devices. We present the computation timings of our dense representation for our CUDA and our OpenMP implementation in Table 1. They can be regarded as pre-processing steps for the interactive navigation. Our CUDA implementation maximally requires around a minute for a whole time series while our CPU implementation is approximately one order of magnitude slower (in our prototype, we only have a CPU implementation for the Mapping). With this, our approach is fast enough to only induce short delays when changing parameters that require recomputation (e.g. the camera position for 3D data or the transfer function). To further support this usage scenario, results are shown progressively after triggering such a recomputation (this is described below in more detail). From the table, it can be seen that the timings for Mapping are linearly dependent on the number of shapes that we need to process, which ranges between 300 000 and 10 million. Note that apart from cases in which updates to the dense representations are required, navigation itself imposes no significant computational cost and is highly interactive. Timings regarding the detection of overdraw for splitting only depend on the image size and introduce negligible computational overhead, taking 0.001 s on the GPU and 0.03 s on the CPU per check.

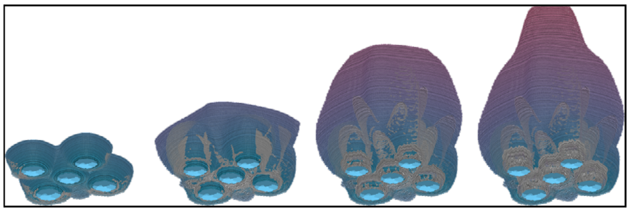


Figure 17: *Progressive computation of our shape representation.*

The primary interaction possibilities considered in this work exclusively use the pre-computed shape information. However, our implementation also supports progressive computation to adequately handle streaming data or changes to the camera position. For this, the operations discussed in Sections 3 and 4 are implemented in different CPU threads to yield a responsive system at any time (denoted below as *representation thread* and *interaction thread*, respectively). The *interaction thread* is the master thread, taking care of user interaction and checking whether there are new (progressive) computation results available from the *representation thread*. When available, the *representation thread* and the *interaction thread* make use of different GPUs. The *representation thread* processes each time step separately and progressively combines the results to an updated version of our representation. This is facilitated by the fact that no particular ordering is required when combining the results of different time steps (cf. Section 3.3). In our implementation, we iteratively integrate new time steps with a batch size of five. We use double buffering, that is, updates to the dense representation are written to alternating buffers. Progressive updates throughout the time series are exemplified in Figure 17 by means of the 5Jets data set.

5.7. Discussion, Limitations and Future Work

The goals of our approach are to provide an overview and support the exploration of moving and transforming shapes. In particular, it is designed for the analysis of spatially sparse data, that is, depicted events do not fully cover the complete image at once over a large number of time steps. As demonstrated earlier in this work, our approach is well suited for analyzing moving droplets, laser pulses, waves and areas of high temperature or velocity magnitude. Still, despite our shape abstraction via forward-facing contours (cf. Section 3.1), temporal overlap still cannot be avoided completely in general. It occurs to a varying degree in all of the examples considered in this work, especially when dealing with data sets with many small and complex overlapping shapes — like the λ_2 data set (Figure 15).

We employ various means in our work to address temporal overlap. First, we indicate overdraw in our representation via reduced colour saturation. During user exploration, temporal persistence of the selected time step in the course of pointer movement allows for smooth, continuous navigation in time. Overlap regions can further be investigated interactively by switching between involved time steps (e.g. Figure 9c and d). For time step selection when hovering over a cell c, we extract maxima in the sequence of weights $\mathbf{c}^t{}_\omega$ over time (cf. Section 4). Note that in our implementation, we use a comparably simple approach for determining extrema that nevertheless proved to work well in our experiments. However, more elaborate and robust approaches for determining extrema could be used as a drop-in replacement (e.g. [VMB13]).

Although these measures work well for dealing with limited spatio-temporal overdraw, they may be insufficient in more challenging scenarios (e.g. as demonstrated at the example of the λ_2 data set in Figure 15a). In such cases, temporal splitting of representations can be effective in preventing overlap in the representation in the first place as discussed in Section 5.4 (e.g. Figure 7). However, there are also potential shortcomings. First of all, this approach introduces parameters with which a user defines the extent of overdraw that is still acceptable. Nonetheless, this downside of parameter dependence is mitigated by the fact that the involved computations for temporal split and rendering are comparably cheap, and therefore parameter adjustments can quickly be reflected in the representations (the computationally expensive Shape Walk is not affected by the split, and accordingly no recomputation is required in this regard when parameters are changed). However, there is a trade-off between limiting overdraw on the hand — potentially resulting in heavily partitioned individual representations, for example, Figure 15m-f — and yielding a concise aggregated overview with few representations on the other.

Although we concentrate on cases exhibiting clear shape boundaries in this work, note that our approach is generic in that it conceptually works directly with smooth and less distinct boundaries as well. Technically speaking, there are no restrictions concerning contours regarding strength $\mathbf{c}_\omega$ (cf. Algorithm 1), length, their total number or mutual similarities in terms of shape or location. However, the presence of softer boundary areas eventually results in more shapes being extracted. This not only increases computation time, but potentially also adds to the issue of temporal overlap. In addition, this might also impede a meaningful selection of contours during temporal navigation. A detailed analysis of scenarios involving less distinct boundaries will be required in future work to closely investigate respective characteristics. Potentially, dedicated extensions could be developed to further improve the expressiveness of the results in these scenarios, for example, by locally clustering and condensing contours. Overall, temporal overlap is not only challenging for our approach discussed in this paper, but generally techniques aiming towards expressive spatio-temporal overviews need to cope with respective visual clutter and occlusion (e.g. [KH13]).

Besides simulation data and experiments that are directly generated in 2D, we also project data that are originally in 3D into the 2D image space. Adjusting the projection parameters (e.g. camera position) can be done fairly quickly due to our efficient implementation, exploiting the significant parallelism involved and supporting progressive updates (Section 5.6). However, we do not preserve depth-related semantics for 3D data in our implementation. As a result, disconnected objects that overlap in the projection are detected as a single shape. In particular, this has an impact on shape classification. In future work, this could be mitigated by further taking depth information into account during Shape Walk. We also consider a fairly simply shape descriptor in our implementation. Although it proved to be quite effective in our experiments, we also plan to study the impact of using other variants in future work.

Our prototype implementation of progressive computation starts from scratch once any parameter is altered that has an impact on the dense shape visualization. To save compute time, we could distinguish between different types of changes and only recompute what is strictly necessary. Finally, we further plan to provide a CUDA-based implementation of Shape Mapping.

6. Conclusion

In this paper, we introduce an approach for the dense visualization and exploration of moving and deforming shapes from scientific experiments and simulations. We create an image space representation by convolving a noise texture along shape contours, in a process that is conceptually similar to LIC. We not only indicate spatial structure via texture, but additionally use colour to depict time or classes of shapes. For shapes, a customized colour map is generated automatically for the provided input data. Our temporally dense representation fully summarizes the evolution of shapes, and with this serves as a basis for interactive user navigation in combination with traditional renderings. We further provide means to adaptively split the representation in time to deal with spatio-temporal overlap. Our efficient implementation quickly generates our representation, and further supports progressive computation. We demonstrate the performance and utility of our approach using 2D and 3D scalar fields from experiments and simulations, analyzing moving droplets, laser pulses, waves and areas of high temperature or velocity magnitude.

In future work, we aim to further investigate different directions to handle extensive spatio-temporal overlap in general, and look into scenarios involving less distinct shape boundaries in particular. Beyond this, we plan to integrate automatic analysis approaches to indicate interesting events to the user, additionally consider depth information when analyzing 3D data, and further improve the efficiency of our implementation.

Acknowledgement

We thank the German Research Foundation (DFG) for funding this work within SFB 1313, Project Number 327154368.

Open access funding enabled and organized by Projekt DEAL.

References

[BCD*] Borgo R., Chen M., Daubney B., Grundy E., Heidemann G., Höferlin B., Höferlin M., Leitte H., Weiskopf D., Xie X.: State of the art report on video-based graphics and video visualization. *Computer Graphics Forum 31*, 8 (2008), 2450–2477.

[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*, Cani M.-P., Ganovelli F., (Eds.). Aire-la-Ville, Switzerland: The Eurographics Association (2012), 75–94.

[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 36*, 8 (2016), 36–61. http://doi.org/10.1111/cgf.12804.

[BVMG08] Balabanian J.-P., Viola I., Möller T., Gröller E.: Temporal styles for time-varying volume data. In *Proceedings of the Fourth International Symposium on 3D Data Processing, Visualization and Transmission* (2008), Gumhold S., Kosecka J., Staadt O., (Eds.), ACM, Atlanta, GA, pp. 81–89.

[CL93] Cabral B., Leedom L. C.: Imaging vector fields using line integral convolution. In, *SIGGRAPH '93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1993), ACM, pp. 263–270.

[CM10] Correa C. D., Ma K.-L.: Dynamic video narratives. *ACM Transactions on Graphics 29*, 4 (2010). http://doi.org/10.1145/1778765.1778825.

[DRB*08] Dragicevic P., Ramos G., Bibliowitcz J., Nowrouzezahrai D., Balakrishnan R., Singh K.: Video browsing by direct manipulation. In CHI '08: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2008), ACM, pp. 237–246.

[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* (2007), Montréal, Québec, Canada, pp. 51–55.

[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 (2016), 508–517.

[FE16] Frey S., Ertl T.: Flow-based temporal selection for interactive volume visualization. *Computer Graphics Forum 36*, 8 (2016), 153–165.

[FE17a] Frey S., Ertl T.: Fast flow-based distance quantification and interpolation for high-resolution density distributions. In *EG 2017 - Short Papers* (2017), The Eurographics Association.

[FE17b] Frey S., Ertl T.: Progressive direct volume-to-volume transformation. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 921–930.

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

[Fre18] Frey S.: Spatio-Temporal contours from deep volume raycasting. *Computer Graphics Forum (EuroVis 2018)* (2018). http://doi.org/10.1111/cgf.13438.

[GGC*08] Goldman D. B., Gonterman C., Curless B., Salesin D., Seitz S. M.: Video object annotation, navigation, and composition. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2008), Association for Computing Machinery, pp. 3–12.

[GKV*07] Girgensohn A., Kimber D., Vaughan J., Yang T., Shipman F., Turner T., Rieffel E., Wilcox L., Chen F., Dunnigan T.: Dots: Support for effective video surveillance. In *MM '07: Proceedings of the 15th ACM International Conference on Multimedia* (New York, NY, USA, 2007), ACM, pp. 423–432.

[HKH*12] Höferlin M., Kurzhals K., Höferlin B., Heidemann G., Weiskopf D.: Evaluation of fast-forward video visualization. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2095–2103.

[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 (2009), 709–718.

[JH95] Jeong J., Hussain F.: On the identification of a vortex. *Journal of Fluid Mechanics 285*, (1995), 69–94.

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

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

[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.

[KDG*07] Kimber D., Dunnigan T., Girgensohn A., Shipman F., Turner T., Yang T.: Trailblazing: Video playback control by direct object manipulation. In *2007 IEEE International Conference on Multimedia and Expo* (2007), IEEE Computer Society, pp. 1015–1018.

[KH13] Kehrer J., Hauser H.: Visualization and visual analysis of multi-faceted scientific data: A survey. *IEEE Transactions on Visualization and Computer Graphics 19* (2013), 495–513.

[Koh98] Kohonen T.: The self-organizing map. *Neurocomputing 21*, 1 (1998), 1–6.

[Kov15] Kovesi P.: Good colour maps: How to design them. Preprint, 2015, https://arxiv.org/abs/1509.03700.

[KWLB08] Karrer T., Weiss M., Lee E., Borchers J.: Dragon: A direct manipulation interface for frame-accurate in-scene video navigation. In *CHI '08: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2008), ACM, pp. 247–250.

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

[LS09a] Lee T.-Y., Shen H.-W.: Visualization and exploration of temporal trend relationships in multivariate time-varying data. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1359–1366.

[LS09b] Lee T.-Y., Shen H.-W.: Visualizing time-varying features with TAC-based distance fields. In *PacificVis '09: IEEE Pacific Visualization Symposium 2009* (Beijing, China, 2009), pp. 1–8.

[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.

[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.

[MSRVH06] Meyer-Spradow J., Ropinski T., Vahrenhold J., Hinrichs K. H.: Illustrating dynamics of time-varying volume datasets in static images. In *Vision, Modeling, and Visualization* (2006). Eurographics Association, Aire-la-Ville, Switzerland, pp. 281–288.

[NNL14] Nguyen C., Niu Y., Liu F.: Direct manipulation video navigation on touch screens. In *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices* (New York, NY, USA, 2014), ACM, pp. 273–282.

[NTN15] Narayanan V., Thomas D. M., Natarajan V.: Distance between extremum graphs. In *IEEE Pacific Visualization Symposium* (2015), pp. 263–270.

[OFCD02] Osada R., Funkhouser T., Chazelle B., Dobkin D.: Shape distributions. *ACM Transactions on Graphics 21*, 4 (2002), 807–832.

[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.

[SW97] Silver D., Wang X.: Tracking and visualizing turbulent 3D features. *IEEE Transactions on Visualization and Computer Graphics 3*, 2 (1997), 129–141.

[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 Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1475–1482.

[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.

[VMB13] Vemulapalli P. K., Monga V., Brennan S. N.: Robust extrema features for time-series data analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 6 (2013), 1464–1479.

[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 Transactions on Graphics 32*, 4 (2013), 44:1–44:8.

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

[WDC*07] Weber G., Dillard S., Carr H., Pascucci V., Hamann B.: Topology-controlled volume rendering. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (2007), 330–341.

[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.

[WYM08] Wang C., Yu H., Ma K.-L.: Importance-driven time-varying data visualization. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1547–1554.

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.

Data Video S1

Data Video S2

Data S3