# Result-Driven Exploration of Simulation Parameter Spaces for Visual Effects Design

Stefan Bruckner and Torsten Möller, *Member, IEEE*

**Abstract**—Graphics artists commonly employ physically-based simulation for the generation of effects such as smoke, explosions, and similar phenomena. The task of finding the correct parameters for a desired result, however, is difficult and time-consuming as current tools provide little to no guidance. In this paper, we present a new approach for the visual exploration of such parameter spaces. Given a three-dimensional scene description, we utilize sampling and spatio-temporal clustering techniques to generate a concise overview of the achievable variations and their temporal evolution. Our visualization system then allows the user to explore the simulation space in a goal-oriented manner. Animation sequences with a set of desired characteristics can be composed using a novel search-by-example approach and interactive direct volume rendering is employed to provide instant visual feedback. A user study was performed to evaluate the applicability of our system in production use.

**Index Terms**—Visual exploration, visual effects, clustering, time-dependent volume data.

## 1 Introduction

Physically-based simulation is gaining increasing popularity for generating realistic animations of water, smoke, explosions, and related phenomena using computer graphics. Common modeling and animation software packages include built-in fluid dynamics simulators or offer this functionality via add-on modules. These existing tools frequently allow the user to modify the simulation parameters via standard controls such as sliders or numeric input fields. It is difficult, however, to predict the influence of changing one or several of these values. Depending on the exact scene setup, effects may be global or remain rather localized, both in space and time. Even small changes can dramatically affect the appearance of the resulting animation. Graphics artists, who aim to produce a particular visual result, therefore typically have to resort to a cumbersome and time-consuming trial-and-error approach. Moreover, as the simulation process is computationally expensive, interactive visual feedback is frequently not available. While recent advances in real-time fluid simulation help by reducing the simulation time [32, 41], the underlying problem remains: there is virtually no guidance in exploring a vast parameter space.

In this paper, we present a result-driven visual approach to navigate through this parameter space tailored to the requirements of graphics artists. Unlike scientists and engineers, who usually seek to understand and analyze the underlying physical phenomenon, these users are primarily interested in controlling the simulation in order to approximate a particular artistic vision. To facilitate this task, we sample the parameter space and apply clustering techniques in an effort to identify the characteristic spatio-temporal variations of the resulting simulations. The results of this process are presented to the user in an interactive visual exploration environment, which combines three-dimensional animated views with an abstracted representation of the identified spatio-temporal clusters. The user can interactively navigate through the space of simulations to find sequences with the desired characteristics using intuitive visual query facilities.

The main contributions of this paper can be summarized as follows: Firstly, we target an application area which, to the best of our knowledge, has not been explored before. Physically-based simulations have become a mainstay in the animation community and visualization tools designed to control the specification of their parameters can help to make the design process considerable less labor intensive. We also present a novel approach for clustering time-dependent volume data generated by sampling a high-dimensional parameter space. Furthermore, the paper describes new visualization and interaction techniques for volumetric time sequences designed to meet the requirements of graphics artists. Finally, we present a user study performed to evaluate the practical applicability of our approach to visual effects design.

• *Stefan Bruckner and Torsten Möller are with GrUVi (Graphics, Usability, and Visualization Lab) at Simon Fraser University, Burnaby, Canada, E-mail: {sbruckne,torsten}@cs.sfu.ca.*

*Manuscript received 31 March 2010; accepted 1 August 2010; posted online 24 October 2010; mailed on 16 October 2010.*
*For information on obtaining reprints of this article, please send email to: tvcg@computer.org.*

## 2 Related Work

The visualization of general time-oriented data is an extensive field of research and Aigner et al. [1] as well as Andrienko et al. [5] provide comprehensive surveys. Our work focuses on the visualization of time-varying volume data, a topic which has been intensively studied in the context of science and engineering data [24]. In many cases, the user is interested in tracking certain features over time which can be difficult in animations of complex data. One approach is to consider the time series as a four-dimensional scalar field. Hanson and Heng [17] introduced general techniques for visualizing surfaces and volumes embedded in four dimensions and developed a 4D illumination model for this purpose. The HyperSlice method presented by van Wijk and van Liere [34] uses a matrix of orthogonal 2D slices as the basic visual representation of a multi-dimensional function. Woodring et al. [40] proposed an intuitive user interface for specifying arbitrary hyperplanes in 4D. After applying slicing or projection, the resulting volume can be displayed using standard techniques. Chronovolumes, presented by Woodring and Shen [37], use integration through time to produce a single volume that captures the essence of multiple time steps in a sequence. A further approach by Woodring and Shen [38] employs different operators to combine multiple volumes. While these methods are useful for detailed analysis and comparison, the resulting visualization can be quite abstract and difficult to grasp.

An alternative approach is to interpret the temporal progression of the data values at each point in space as a one-dimensional function referred to as a time-activity curve [14]. These curves can be used to identify spatial regions with certain properties. Muigg et al. [27] presented techniques for the visualization of a large set of these function graphs for applications such as breast tumor diagnosis. Woodring and Shen [39] applied clustering to time-activity curves to identify similar regions in space. The approach by Lee and Shen [22] attempts to identify temporal trends and models them as a state machine of trend sequence. A further approach for characterizing time-dependent volume data are time histograms which represent information about the frequency of occurrence for each data value and time. Akiba et al. [3] used time histograms to assist in the specification of transfer functions across multiple time steps. For the visualization of multi-variate time-dependent data, Akiba and Ma [2] also proposed the combination of

1077-2626/10/$26.00 © 2010 IEEE Published by the IEEE Computer Society

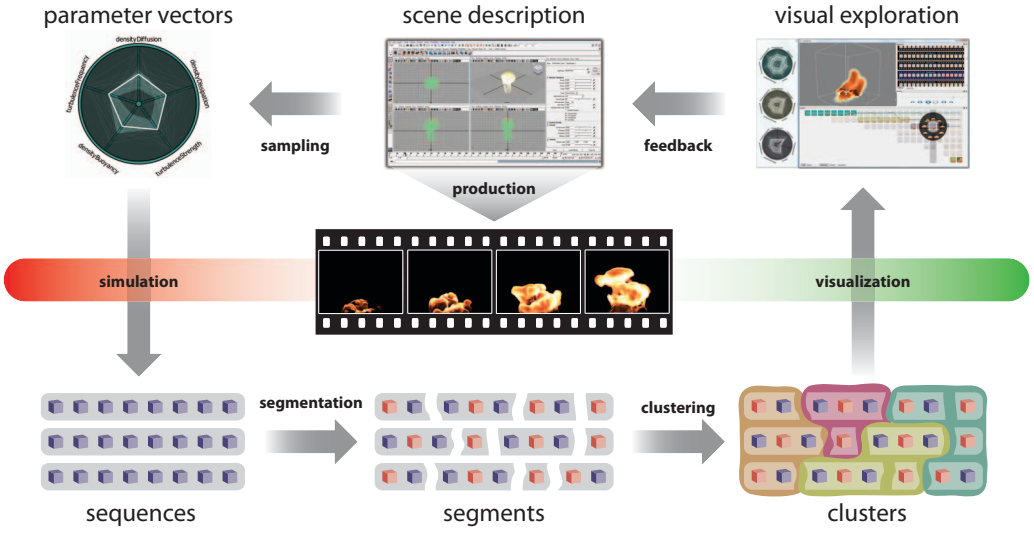



Fig. 1. Conceptual overview of our visualization system. The process starts from a scene description which defines the basic simulation scenario. Sampling generates a set of parameter vectors which are used to control the simulation process. The resulting sequences are then split into multiple short segments and clustering is applied to group these segments. The results can be interactively explored to find the desired parameter settings for the final animation.

time histograms and parallel coordinates. Our goal differs from these methods in that we do not attempt to track features over time or characterize the behavior of different regions. Instead, we want to globally investigate the similarities and characteristic variations between multiple volumetric time series.

Thus, while based on time-dependent volume data, our approach bears many similarities to methods from video processing and content retrieval. In order to overcome the sequential and time-consuming process of viewing video a noticeable amount of effort has been made to devise methods for analyzing and abstracting video data automatically [33]. A first step in many approaches is shot detection, i.e., partitioning the video into multiple series of interrelated consecutive frames. Hanjalic [15] provides a detailed overview of different methods employed for this purpose. A further step may involve clustering of these shots to extract a compact representation of the video in the form of representative key frames or preview sequences [16, 28]. Within the visualization community, Daniel and Chen [10] proposed the use of volume rendering to present summaries of video sequences. Our work draws inspiration from video analysis and abstraction methods and has many related goals such as the easy visual retrieval of data.

Dimensionality reduction and clustering are commonly employed for gaining insight into high-dimensional parameter spaces [6,21]. Our input parameter space is also multi-dimensional, but we apply clustering to characterize the output space of simulations to extract information about visual variations over time. Furthermore, visualization and query techniques for interacting with complex sets of temporal data such as ThemeRiver [19], TimeSearcher [20], and PatternFinder [13] have inspired our work.

Finally, our approach is most closely related to techniques for design space exploration. Ma [23] introduced a visualization system which presents information on how parameter changes affect the result image as an image graph based on data generated during an interactive exploration process. Smith et al. [31] presented methods for navigating through a complex shape space of registered car models using an intuitive direct manipulation interface. The work of Monks et al. [26] discussed a system for acoustic design which applies visualization, simulation, and optimization in a goal-oriented manner. Marks et al. [25] introduced Design Galleries, a general concept for exploring parameter spaces. Our system is founded in their basic methodology of sampling the input space to generate a visual overview. In contrast to the methods presented in this paper, however, their work only discussed static output and did not address the complex issue of time-dependent data.

## 3 Overview

In order to distinguish our goal from that of typical simulation visualization approaches targeted at scientists and engineers, an analogy to biology can be drawn. The genome encodes the set of instructions for building a living organism. The term *genotype* refers to an organism's full hereditary information, even if not expressed. The term *phenotype*, on the other hand, refers to an organism's actual observed properties, such as morphology or behavior. Different genotypes may result in similar phenotypical characteristics during different stages of development. *Cladistics* is the systematic study of organisms based on their genetic relationships, while *phenology* attempts to classify organisms based on overall similarity regardless of their evolutionary relation. Even though most of today's evolutionary biologists favor cladistics, phenetic approaches can prove useful when studying diverse groups of closely-related organisms. Similarly, we want to provide visualization tools to explore the *simulation space*, i.e., our main focus lies in visualizing the variability in observable characteristics of a set of simulations. In this sense, our approach can be considered deliberately phenetic. In contrast, if the primary goal is to analyze the underlying *parameter space*, a cladistic approach is usually more suitable.

A conceptual overview of our system is depicted in Figure 1. We start from a *scene description* generated in a standard modeling/animation software package. It consists of the basic simulation settings, such as duration, geometric setup, and emitter specification. While many artists have developed an intuition which general parameters need to be tuned in order to achieve a certain result, the actual parameter values are highly dependent on the specific nature of the scene. To provide visual guidance in this selection process, our approach begins by randomly sampling a manually selected subset of the parameter space. The sampling process generates a set of *parameter vectors*. For each of these combinations of parameter values, a simulation consisting of multiple time steps in the form of volumetric grids is produced. These *sequences* may exhibit different characteristics at different points in their temporal evolution. For instance, they share the same initial state and, depending on the parameters, can diverge at varying rates. Likewise, multiple simulation sequences may start to converge to similar states as they progress. As a simple example, consider a smoke simulation: Initially smoke will rise, but, depending on the temperature, gravity will cause the smoke particles to fall again at

a certain point in time. In order to capture these kinds of characteristic variations, we evaluate the spatio-temporal similarity of the generated simulations. First, a segmentation step decomposes each simulation sequence into multiple continuous *segments*. A density-based clustering algorithm is then applied to group multiple similar segments into visually distinct phases. The results of this classification process are presented in an easily-understandable layout for interactive *visual exploration*. The user can inspect the different variations and use intuitive interaction tools to find sequences which exhibit the desired spatial and temporal characteristics. The corresponding parameters, or the already generated simulation, can then be used for production of the final animation.

The remainder of this paper details these individual steps and components. Section 4 is devoted to sampling, segmentation, and clustering, while Section 5 focuses on visualization and interaction techniques. Implementation details are discussed in Section 6. Section 7 presents the results of a user study performed to evaluate the suitability of our system for production use. The paper is concluded in Section 8.

## 4 Sampling and Clustering

In this section, we describe the individual processing steps which form the basis of our approach.

### 4.1 Sample Generation

The high computational costs of fluid simulation severely constrain interactive exploration within the authoring environment. In an effort to eliminate the cumbersome trial-and-error process of changing a parameter value, waiting for the result to compute, and then deciding whether the desirable effect has been achieved, we generate random simulation samples in an offline process. While this may seem costly, both in terms of processing time and storage demands, it has the considerable advantage that it can be performed without requiring user intervention, e.g., overnight. Animation studios are typically equipped with render farms, so this setup fits well into the environment of our intended users. Initially, the user chooses a set of $M$ simulation parameters:

$$P = \{p_1, p_2, \ldots, p_M\} \tag{1}$$

where each parameter $p_i \in P$ has an associated range of interest $R_i = [a_i, b_i] \subset \mathbb{R}$. This choice is mostly influenced by the desired effect and the physical interpretation of these parameters. A selected number of $N$ samples of this $M$-dimensional parameter space will be generated. We refer to each combination of simulation parameter values as a *parameter vector* $\mathbf{x} \in \mathbb{R}^M$ :

$$\mathbf{x} = (x_1, x_2, \ldots, x_M) \tag{2}$$

with $x_i \in R_i$. For each parameter vector $\mathbf{x}$, the simulation module then generates a *simulation sequence* $S(\mathbf{x})$ written as a set of $T$ time steps:

$$S(\mathbf{x}) = \{s_1, s_2, \ldots, s_T\} \tag{3}$$

where each $s_i$ is a volumetric grid. Depending on the type of simulation, each grid point may store multiple attributes, such as density, temperature, pressure, etc. For simplicity, the remainder of this paper will focus only on scalar output, but our methods equally apply to multi-channel data. For most common effects density and temperature are simulated and typically mapped to, respectively, opacity and color. The sampling process generates $N$ sequences consisting of $T$ volumes. For most visual effects, simulations will be rather short with $T$ ranging from tens to a few hundred frames. Grid sizes vary depending on the specific effect, but are typically smaller than for other common types of volume data such as medical scans. $N$ should be chosen according to the number of parameters, but is constrained by the simulation cost in terms of processing time and disk space requirements.

In our current implementation, we use unconstrained random sampling as it permits the easy addition of further samples as well as termination of the sampling process at any time. However, to ensure a more uniform coverage of the parameter space alternative schemes such as Latin hypercube sampling may be used instead. One major practical advantage of random sampling is that the exploration of intermediate results is easily possible.

### 4.2 Sequence Segmentation

To facilitate robust clustering as well as to reduce the computational load of the subsequent processing steps, our approach first splits each volumetric time sequence $S$ into multiple short *segments* $S' \subseteq S$ of varying length. It is important to note that, at this point, we are not concerned with identifying overall similarity. Rather, we want to divide each simulation sequence into a smaller number of manageable units which exhibit high similarity and are continuous in time. Here, we draw inspiration from the field of video processing. Many methods for generating an overview of a video clip start by dividing the input into multiple shots by detecting discontinuities [15]. In contrast to these methods, however, our approach groups neighboring time steps as a simulation will in general not exhibit distinct boundaries.

For each sequence $S$, we compute a dissimilarity measure between neighboring time steps of a sequence using the sum of squared intensity differences over all grid points of the corresponding volumes:

$$d_S(t) = \sum_{\mathbf{u}} (s_{t+1}(\mathbf{u}) - s_t(\mathbf{u}))^2 \tag{4}$$

where $s_t(\mathbf{u})$, $s_{t+1}(\mathbf{u})$ are the data values at the three-dimensional grid position $\mathbf{u}$ of two subsequent time steps with $t \in [1, T-1]$. We then use a simple greedy algorithm which merges neighboring time steps based on their dissimilarity. Initially, each time step forms its own segment. The cost value associated with each segment is initialized to zero. We then iteratively merge two neighboring time steps if the dissimilarity at their boundaries added to their individual costs is minimal. The cost of the resulting segment is updated to this sum. This process proceeds at least as long as the number of segments is larger than a specified value for the maximum segment count. After that, the algorithm terminates when the minimum cost exceeds a threshold. The first parameter, the maximum number of segments, is set according to the available computational resources – a larger number of segments will increase the time required for the subsequent clustering step. For the cost threshold, we use the average dissimilarity of the sequence.

The result of this algorithm is a varying number of segments for each simulation sequence. The representative time step $r(S')$ for a segment $S'$ is chosen such that it minimizes the absolute difference between the cumulative dissimilarity of its predecessors and successors within the segment:

$$r(S') = \underset{s_i \in S'}{\arg\min} \left| \sum_{s_j \in S', j<i} d_S(j) - \sum_{s_j \in S', j>i} d_S(j-1) \right| \tag{5}$$

For segments with only two members, the lower time step is chosen. In the subsequent clustering step, all of the individual time steps represented by one such segment are treated as a unit and the representative time step is used in their place.

Figure 2 depicts an example of the sequence segmentation process. A short sequence of 25 time steps is split into 7 segments. The graph shows the dissimilarity $d_S(t)$ for $t \in [1, 24]$ – note that for the last time step $t = 25$ of the sequence, this function is undefined. The highlighted points indicate the chosen representative time steps and the images show renderings of the corresponding volumes. No minimum number of segments was specified and the cost threshold was set to the average dissimilarity.

### 4.3 Density-based Clustering

Having split each simulation sequence into a number of representative segments, we now aim to compare the simulation space on a global level, i.e., we want to identify similar phases or states which may occur at different points within the temporal evolution of each simulation. For this purpose, we employ a density-based clustering approach. In contrast to partitional and hierarchical approaches, density-based clustering uses a local cluster criterion, in which clusters are defined as regions in the data space where the data points are dense, separated from one another by low-density regions. In particular, we employ a variation of the DBSCAN algorithm [12], as it has the ability of discovering clusters with arbitrary shape and does not require the predetermination

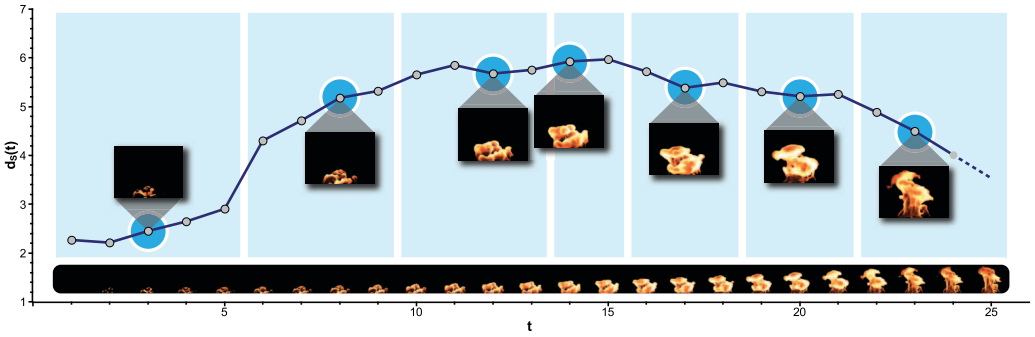



Fig. 2. Sequence segmentation. A sequence of 25 time steps split into 7 segments using our algorithm is shown. The graph depicts the dissimilarity $d_S(t)$ between two subsequent time steps $t$ and $t+1$. The highlighted points indicate the selected representative time steps for each segment.

of the number of clusters. This is advantageous, as it allows us to make minimal assumptions about the similarity relationships in simulation space.

DBSCAN requires two parameters: $\varepsilon$, which defines the maximum distance between two points considered to be neighbors, and $p_{\min}$, the minimum number of points required to form a cluster. The algorithm starts with an arbitrary point that has not been visited. This point's $\varepsilon$-neighborhood is retrieved, and, if it contains sufficiently many points, a cluster is started. Otherwise, the point is marked as noise. This point may later be found to be in a sufficiently-sized $\varepsilon$-environment of a different point and hence still become part of a cluster. If a point is found to be part of a cluster, its $\varepsilon$-neighborhood is also part of that cluster. Thus, all points that are found within the $\varepsilon$-neighborhood are added to its cluster, as is their own $\varepsilon$-neighborhood. This process continues until no further points can be found. Then, a new unvisited point is processed.

In our case each segment, identified by its representative time step, corresponds to one point. For the neighborhood size, which allows DBSCAN to judge local density, however, the point contributes with the number of members of the corresponding segment. This enables the information gathered during sequence segmentation to influence the clustering process. An important choice is the dissimilarity measure employed in the clustering algorithm – as motivated in Section 3, we are primarily interested in visualizing the simulation data in terms of their observable characteristics. Feature-based distance metrics have shown to have many advantages for various clustering tasks and much work has been devoted to developing techniques for extracting features in scalar- as well as vector-valued volume data [30]. However, most of these methods require several parameters and are tailored to specific tasks. Since we want to compare hundreds of volumes, manual parameter selection is not an option (indeed, the main motivation of our work is to simplify parameter specification). Moreover, it would be difficult to define a feature vector which provides a robust basis for comparing simulation time steps generated across the full range of the parameter space. Thus, instead of attempting to extract explicit features, we use a rather simplistic dissimilarity measure based on the sum of squared intensity differences between two volumes $v_1$ and $v_2$:

$$d(v_1,v_2) = w(v_1,v_2)\sum_{\mathbf{u}}(v_1(\mathbf{u})-v_2(\mathbf{u}))^2 \tag{6}$$

with

$$w(v_1,v_2) = 1+\begin{cases}\sqrt{|\mathrm{ti}(v_1)-\mathrm{ti}(v_2)|} & \text{if } \mathrm{si}(v_1)=\mathrm{si}(v_2)\\ 0 & \text{otherwise}\end{cases} \tag{7}$$

where $v_1(\mathbf{u})$ and $v_2(\mathbf{u})$ are the data values at the three-dimensional grid position $\mathbf{u}$, $\mathrm{ti}(v_1)$, $\mathrm{ti}(v_2)$ are the time step indices, and $\mathrm{si}(v_1)$, $\mathrm{si}(v_2)$ are the sequence identifiers of, respectively, the volumes $v_1$ and $v_2$. The additional weight increases the dissimilarity of time steps within one sequence based on their temporal differences. This allows the clustering algorithm to group similar temporal progressions between different sequences even if they are not entirely synchronous. The approach is related to the measures proposed by Birant and Kut [9] even though they focus on geographic time series data. The choice of the sum of squared differences as the basis for our measure is motivated by its frequent use in image registration tasks [35]. While other measures, such as mutual information, may perform better they also come at significantly higher computational costs.

The clustering step then proceeds as follows: First, we compute the dissimilarity matrix by comparing each pair of segment representatives. Next, for each segment representative, a neighborhood index is generated by sorting the dissimilarity values in ascending order. The DBSCAN algorithm is then executed resulting in a number of clusters. If a segment representative is part of a cluster, all members of the corresponding segment are assumed to share this association.

For specifying the parameters of the algorithm, we use a simple heuristic [12]. The minimum number of points $p_{\min}$ is set to:

$$p_{\min} = \ln\left(\frac{1}{\overline{|S'|}}NT\right) \tag{8}$$

where $\overline{|S'|}$ is the average number of members in a segment, $N$ is the number of sequences, and $T$ is the number of time steps per sequence. We set $\varepsilon$ to the average dissimilarity between all segments. In our experiments these values have shown to be robust defaults. However, as the time required for executing the actual clustering algorithm once the dissimilarity matrix has been computed is negligible, these values can also be adjusted easily if an unusually low or high number of clusters is detected.

In addition to references to its members, each cluster stores the following additional information computed after the clustering algorithm has completed:

**Cluster medoid** – the cluster member which minimizes the average dissimilarity to all other members in the cluster.

**Sequence range** – the set of sequence identifiers which have at least one member in the cluster.

**Temporal range** – the set of time step indices covered by the members of the cluster.

**Temporal medoids** – for each distinct time step index contained in the cluster, the member which minimizes the average dissimilarity to all cluster members of the same time step.

In the following section we discuss how this information is used to generate a compact representation of the simulation space's temporal evolution.

## 5 Interactive Exploration

Having identified spatio-temporal clusters in the set of sampled simulations, we want to present this information, together with the original

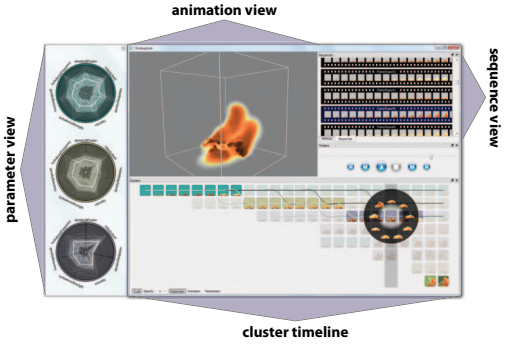



Fig. 3. Screenshot of our interactive exploration environment.

simulation sequences, in an easily-understandable manner. The general layout of our interactive visualization system is shown in Figure 3. The user interface consists of several different linked views. The *animation view* shows a volume rendering of the currently selected sequence and is controlled by a standard time slider. The *sequence view* allows the user to browse through all available simulation sequences. The *cluster timeline* is the main element for our application – it gives an overview of the visual variations across the simulated sequences over their temporal range and allows the user to search for sequences with particular characteristics. Finally, the *parameter view* provides a visualization of the parameter space variations for a selected set of clusters. All views are linked, so selecting a particular sequence in the sequence view, for example, will update the animation view and highlight the corresponding elements in the other views.

### 5.1 Animation View

As the final result of the process assisted by our visualization system is an animation sequence, it is important to provide an interactive preview. The animation view depicts a volume rendering of the currently selected sequence. It allows viewpoint manipulation and playback using a standard time slider and animation controls. Rendering of time-dependent volume data is an active area of research and many powerful techniques capable of dealing with large data sets have been presented [24]. As this is not the focus of our paper, we will only briefly describe our setup. Our system features two different volume renderers: An emission/absorption ray caster implemented in CUDA, and a slice-based renderer which supports self-shadowing and scattering approximations based on a conical phase function. The latter renderer uses OpenGL since the slice-by-slice processing scheme required by its illumination model performs significantly better in OpenGL than a comparable CUDA implementation as it can exploit fixed-function GPU operations. The CUDA renderer offers better overall performance, while the superior optical model of the OpenGL renderer provides higher fidelity. The user can switch between these two renderers at runtime.

### 5.2 Sequence View

The sequence view provides a simple overview of all simulated sequences by depicting a "film strip" of their time steps. One of its purposes is to allow the user to establish a mental model of the visualization process. The sequence view displays all simulation sequences. Hence, the animation view which only displays a single time step at a time, and the cluster timeline, which provides a summarized and abstracted view of the sequences' temporal progression, can be interpreted as filtered representations of the data. The sequence view is linked to the other views, so whenever the current sequence is changed it is scrolled into view and highlighted. While it would also be possible to use the results of sequence segmentation for selecting the depicted images, we instead choose to uniformly divide the time range so the chosen time steps are the same for all sequences resulting in a more traditional presentation familiar from common animation and video processing software. Depending on the width of the view, the number of images is adjusted to fill the available space. The displayed images are live thumbnails, i.e., they are updated whenever settings such as the current viewpoint change. Image generation is performed in the background using our CUDA volume renderer and the resulting thumbnails are cached in main memory.

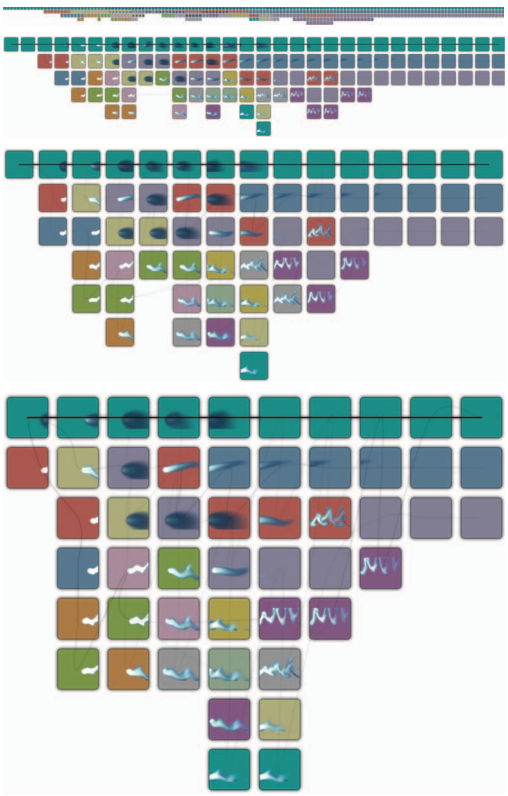
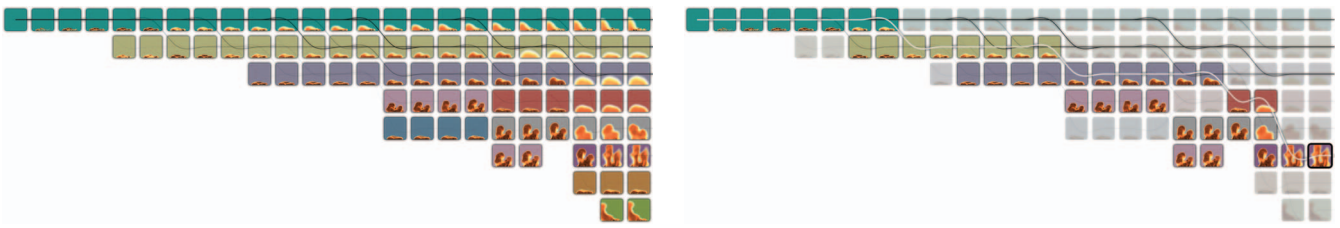


Fig. 4. Cluster timeline at different temporal compression levels. The depicted time interval sizes are, from top to bottom: 1, 5, 10, and 15. The sequences in this set of 128 simulations of a bullet passing through a medium have 150 time steps each.

### 5.3 Cluster Timeline

The cluster timeline provides a concise overview of the visual variations over the duration of the simulation while summarizing the similarities between different sequences and represents the key component of our interface. The general idea behind this visualization is to consider the identified clusters as distinct phases in the temporal evolution of a simulation. Multiple sequences may enter a particular phase at different points in time as they progress. Since, depending on the nature of the simulation, potentially many of these phases exist, the generated layout should be compact. Our dissimilarity measure is successful in grouping similar segments, but since it is not feature-based the inter-cluster distance is less informative and we choose not to visually encode it which provides more freedom in designing a compact layout. While our experiments have shown that the identified clusters tend to cover continuous time ranges, we do not explicitly enforce temporal continuity so the visualization algorithm needs to be capable of handling discontinuous clusters as well. Finally, as artists are used to dealing with linear time scales in their standard tools, we choose not to distort the time axis.

Based on these general guidelines, we developed a simple layout algorithm which visualizes the temporal distribution of clusters and their membership relationships. The cluster layout is generated in the following manner: Initially, all clusters $C$ are assigned a global rank $r_G(C)$ defined as the product of the total number of time steps covered by the cluster and the number of distinct sequences it contains. This means that clusters which cover a large temporal range and/or include



(a) (b)

Fig. 5. Interaction with the cluster timeline. (a) No selection has been made, so all cluster items remain active. (b) A cluster item has been selected, the corresponding sequence path is highlighted, and unconnected items are dimmed. The depicted data set is a fire effect consisting of 128 sequences, each with 25 time steps.

many different sequences will be ranked higher. Each cluster is assigned a color using one of the ColorBrewer's [18] qualitative color schemes. As it is generally advised against attempting to visually encode too many classes using color, the assigned cluster colors may not be unique. If the number of total clusters exceeds the number of colors in the scheme (the maximum is 12), we maximize the temporal difference between clusters which are assigned the same color.

*Cluster items* are then positioned on the canvas by traversing the temporal simulation range. One cluster item represents a subset of the cluster's members which share a common time interval. Thus, the number of associated cluster items varies depending on the temporal extent of a cluster. For every non-overlapping time interval $[t_s,t_e)$ and every cluster $C$, the number $r_T(C,t_s,t_e)$ of cluster members which are contained in the interval is determined. All clusters where this number is greater than zero are sorted in descending order according to the product of $r_G(C)$ and $r_T(C,t_s,t_e)$ and one item is created for each of these clusters. The horizontal position of the item is determined by the current interval $[t_s,t_e)$ while the vertical position (from top to bottom) corresponds to the sorting order. The visual representation of a cluster item consists of a background rectangle in the cluster color and a live thumbnail image depicting a rendering of the cluster's temporal medoid for the current interval, i.e., the cluster member which minimizes the average dissimilarity to all other members within the same time interval.

To provide an overview of the temporal progression of the individual simulation sequences, a *sequence path* is generated for every sequence. The path represents the progression of cluster memberships of a sequence over time and is displayed as a cubic spline connecting the centers of all cluster items the sequence is a member of. It is drawn using an opacity based on the total number of sequences which enables the identification of membership patterns that occur more frequently. The path of the current sequence, i.e., the one that is displayed in the animation view and selected in the sequence view, is emphasized and drawn with full opacity on top of all other paths. The resulting layout depicts, for every time interval, the possible variations identified in the clustering process. Due to the influence of temporal range on the sorting order, clusters covering many time steps appear first. The user can control the size of the interval interactively using a slider. When changing the interval size, the size of the cluster items on screen remains the same, but the time axis gets compressed. Cluster items which represent the same cluster merge, while those of different clusters stack on top of each other according to their rank. At the highest zoom level, i.e., the interval spans the entire vertical range, all clusters are listed vertically according to their rank. This sort of temporal compression enables viewing of long sets of sequences without scrolling, while preserving their salient features and variations. As a sequence path may connect several cluster items at the same horizontal position for large intervals, we only connect them to the highest ranked cluster item in these cases.

Figure 4 shows an example of the cluster timeline at different temporal compression levels. The sequences in this set of 128 simulations have 150 time steps each. The depicted time interval sizes are, from top to bottom: 1, 5, 10, and 15. Note that on screen the cluster items always have the same size and the view scrolls horizontally.

#### 5.3.1 Search-By-Example

To enable result-driven exploration, the user can interact with the cluster timeline. Selecting a cluster item will highlight all other cluster items which have members that connect to it, i.e., there is a sequence which is a member of both clusters within the time interval of the item. Multiple items can be selected thereby further filtering the view. Whenever such a selection is made, the current sequence displayed in the animation and sequence views is instantly updated to the best match of the query and the corresponding sequence path is highlighted in the cluster timeline. When the selection is modified, candidate sequences, i.e., those which connect to all selected cluster items, are ranked by the number of their time steps which are members of the corresponding clusters. Of those sequences with the maximum number of overlapping time steps, the one which minimizes the dissimilarity to the temporal medoids of all selected cluster items is chosen. This strategy prefers sequences with longer membership times in the clusters corresponding to the selected items, i.e., their sequence paths will tend to be more straight.

An example for this type of interaction with the cluster timeline is shown in Figure 5. The cluster timeline for a flame effect simulation is depicted in Figure 5 (a). When a cluster item is selected, as shown in Figure 5 (b), all items which share no connection with the selected item are dimmed. The sequence path of the best query match is emphasized. The cluster items which remain active indicate the possible variations which share a similar end state.

This intuitive visual query metaphor enables quick identification of sequences with the desired spatial and temporal properties, or alternatively, that there are no such sequences. The linked animation view additionally adds to the flexibility of this approach for finding desired simulation characteristics. When selecting a cluster item, the sequence depicted in the animation view changes according to the best match of the query, but the time position remains unchanged and can be controlled independently by the time slider. Only when an item is double-clicked, the time slider is moved to the start of its interval. This enables the user to, for instance, quickly switch between different variations early in the temporal progression and get a three-dimensional view of their evolution at a later point in time.

To explore the variations within a cluster, a radial context menu can be opened by right-clicking a cluster item. The menu depicts the nearest neighbors of the cluster item's temporal medoid which are part of the same cluster, but not necessarily at the same time step. By clicking on one of the displayed thumbnails, the corresponding sequence is selected. This enables navigation within a cluster. The context menu is visible in the screenshot shown in Figure 3.

#### 5.3.2 Sequence Blending

Sometimes, a particular desired temporal progression may not occur in the set of simulations. This can be due to the limited number of samples, but it may also be the case that it is physically impossible given the chosen set of simulation parameters. Nonetheless, artists will often sacrifice physical plausibility for achieving a desired result. Animation packages usually include functionality to combine different simulation runs. In our system, we allow the user the possibility to specify blending between sequences directly in the cluster timeline.

A cluster item can be marked as a key frame, i.e., the best matching sequence for the item, as described previously, will be displayed until the end of the cluster item's time interval. If a further key frame is selected, a transition between the two sequences will occur in the animation view within the time interval between the two cluster items. Blending is performed in volume space using on-the-fly interpolation between the corresponding time steps of both sequences based on a user-specified easing curve. An extension of these simple animation facilities using approaches such as those presented by Wohlfart and Hauser [36] or recent work by Akiba et al. [4] could also be an interesting direction for further research.

### 5.4 Parameter View

While, as initially stated, it is not our primary goal to facilitate detailed analysis of the parameter space, it is still useful to provide an overview of the parameter variations within a cluster. For this purpose, we employ circular parallel-coordinate plots inspired by DataRoses as proposed by Elmqvist et al. [11]. For each selected cluster, the parameter view shows a star plot layout where each of the simulation parameters corresponds to one of the equiangular axes. As it is common to facilitate side-by-side comparisons, all star plots use the same scaling with the minimum of the parameter range at the center and the maximum located at the radius of the circle for each axis. Since we want to visualize the parameter distribution in relation to its visual manifestation, the depicted parameter vectors are weighted indirectly proportional to the dissimilarity of the corresponding segment to the cluster medoid. The weights are scaled such that the medoid is assigned a value of one and the member with the highest dissimilarity to the medoid receives a weight of $|C|^{-1}$ where $|C|$ is the number of cluster members. The parameter vector for each cluster member is then depicted as a polygon with an opacity proportional to its weight. The weighted mean of parameter vectors within the cluster is shown as thick white polygon with full opacity. Additionally, the region enclosed by the first and third weighted quartile is highlighted.

Figure 6 depicts the parameter view for three clusters of a flame effect together with a rendering of the final time step of a cluster member. Note how a correspondence between lower dissipation and a more typical fire-like appearance is indicated when observing the distribution of parameter vectors within each star plot. The corresponding cluster timeline is shown in Figure 5.

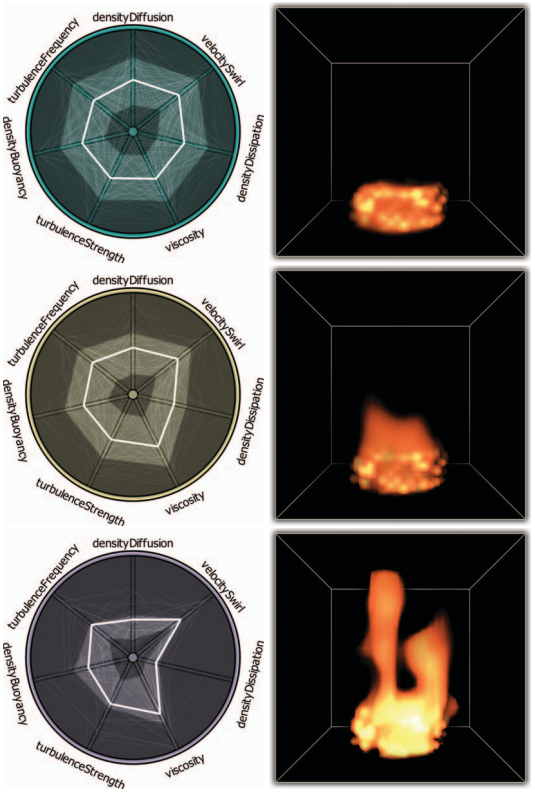



Fig. 6. Parameter view for three clusters of a set of flame simulations together with renderings of representative cluster members.

## 6 Implementation

Our system was implemented in C++ using the Qt cross-platform application framework and consists of two basic parts: the stand-alone visualization application and a processing module. The processing module was implemented as a plugin-in for Autodesk Maya. The sampling process can be initiated using a command in Maya's scripting language MEL or can be bound to a user-interface element. Both modules communicate via sockets, and can therefore be used over the network.

In order to provide a fully responsive interactive experience without delays, we heavily rely on parallelism at several different levels. Even if individual volume dimensions may be comparably small (they usually do not exceed dimensions of $128^3$), the typical amount of data still consists of several hundreds of these volumes and cannot be held in main memory. We employ NVidia's CUDA GPU computing platform, which allows synchronous execution of GPU processing tasks and memory transfers. Both can also be performed concurrently with CPU processing using CUDA's stream concept. This high degree of parallelism allows for excellent latency hiding and a fully controllable memory footprint. In our architecture, each individual volume – referred to as a data item – is identified by a unique index. A component which requires access to one or multiple data items places an asynchronous request for the desired index range and continues operation. The data access component collects and schedules these requests according to priority and access pattern. Two memory pools are used to cache data: one in main memory and one on the GPU. When placing a data request, the caller can indicate whether it wants to access the data on the CPU, the GPU, or both and cache replacement is performed accordingly. For example, if the data is only required on the GPU, e.g., for volume rendering, the corresponding slot in the main memory pool can be marked as available for replacement immediately after transfer to GPU memory has been completed. A background thread is responsible for loading data items from hard disk into main memory and, if requested, initiates transfer to GPU memory. For each data item in the requested index range a notification is sent when it is available in the indicated memory pool. Before the requesting component is notified, the data item is marked as locked to prevent cache replacement until processing has finished. The requesting component is responsible for releasing the item as soon as the data is no longer required. Both the CPU and the GPU memory pool use a Least-Recently Used (LRU) cache replacement policy. Requests can also be marked as optional, i.e., they are scheduled whenever no other items are being transferred. In this case, no locking occurs and the caller is not notified of availability. This is useful for prefetching data during animation rendering.

## 7 Evaluation

While we described the capabilities of our system and attempted to illustrate them in static images, it is difficult to fully capture interactive processes in this manner. We therefore refer the reader to the accompanying video for a live demonstration. Table 1 lists information on the depicted data sets as well as simulation and preprocessing times.

The research presented in this paper was motivated by animation professionals who deal with the problem of missing visual guidance in choosing simulation parameters on a daily basis. Their requirements guided our development process. In order to evaluate the functionality, usability, and potential practical impact of our system, as well as to identify areas which require further research, we performed a user study. Based on previous demonstrations to practitioners and our own experience with the system, we had the following hypotheses:

1. Our general approach for visualizing fluid simulations for visual effects design will be considered useful and valuable.

Table 1. Statistics and performance numbers for the processing of two data sets. Timings are given for simulation, segmentation of the simulation sequences, computation of the dissimilarity matrix, and clustering. System configuration: Intel Core 2 Duo 2.53 GHz CPU, 4 GB RAM, NVidia GeForce 9600M GT GPU.

| Data set | Flame | Bullet |
|---|---|---|
| Sequences | 128 | 128 |
| Time steps | 25 | 150 |
| Resolution | $30 \times 30 \times 30$ | $100 \times 40 \times 30$ |
| Disk space | 230 MB | 8.8 GB |
| Segments | 899 | 3183 |
| Clusters | 8 | 22 |
| Simulation | 23 min | 448 min |
| Segmentation | 1 min | 7 min |
| Dissimilarity | 2 min | 76 min |
| Clustering | $< 1$ s | 8 s |

2. The cluster timeline will be considered helpful in exploring the variations in the set of simulations.

3. There will be difficulties in understanding the parameter view and it will be considered less helpful.

The study was performed one participant at a time using the following protocol: Each participant was first asked to fill out a background questionnaire and then received a general verbal introduction into the concepts behind our approach, followed by a live tutorial on how to use the software. Each participant also received a one-page summary of the mouse and keyboard mappings in the application. Next, the participant was asked to perform a list of simple tasks. We employed the think-aloud protocol, i.e., the participants were asked to verbalize their thoughts and actions. We specifically designed the tasks to be open-ended and to rely on the subjective judgement of the user, for example *"Find the simulation sequence that provides, in your judgement, the most realistic appearance"*. No time limit was given and the study participants were encouraged to freely explore all aspects of the application. Screen capture and audio recordings were made to document the user interaction with the system. After completing the tasks, the participants were asked to fill out a post-questionnaire in which they had to rate 25 statements on a 5-point attitude Likert scale. The questionnaire covered general application functionality, suitability and difficulty of tasks, as well as the assessment of individual components (e.g., *"I found the cluster timeline was more helpful in completing the tasks than the sequence view"*). Additionally, it also included the ten items of the System Usability Scale (SUS). This evaluation technique provides a global assessment of overall usability and user satisfaction on a scale ranging from 0 to 100 and has been shown to yield reliable results even for small user groups [7]. Finally, a semi-structured interview consisting of questions on the overall impression as well as several specific topic areas was performed.

Using this protocol, the study was performed on a total of 12 subjects (9 male, 3 female) divided into two groups. Group A consisted of 7 (5 male, 2 female) interactive arts students with moderate to expert knowledge in 3D modeling and animation, but generally less experience with fluid simulation. Group B consisted of 5 (4 male, 1 female) visual effects professionals with several years of expertise in the subject matter who routinely employ fluid simulation in their work. While the targeted duration of an evaluation session was approximately one hour per person, there were considerable variations with some subjects choosing to spend more time on exploring different aspects of the system and/or making extensive comments during the interview.

All participants agreed that the functionality provided by the system was useful. Most subjects found that the cluster timeline provided a good summary of the variations within the set of simulations (10 agree, 2 disagree). Interestingly, while all subjects from group B found that the cluster timeline was useful in completing the tasks, the corresponding scores of group A showed more variability (4 agree, 1 disagree, 2 neutral). Similarly, most subjects from group B (4 agree, 1 disagree) thought that the cluster timeline was more helpful than the sequence view in completing the tasks, while the response of group A was less uniform (3 agree, 2 disagree, 2 neural). Only a minority of the subjects found the parameter view useful (5 agree, 3 disagree, 4 neutral). In general, the selected tasks were considered to be easy to understand (10 agree, 2 disagree), appropriate for assessing the functionality of the system (8 agree, 1 disagree, 3 neutral), and relatively easy to complete (8 agree, 1 disagree, 3 neutral).

Most subjects found that the system was generally easy to use (10 agree, 1 disagree, 1 neutral). However, the average SUS score of 62.7 with a standard deviation of 13.1 also indicates that our current prototype needs additional work on user interface and interaction design. According to the work of Bangor et al. [8], this corresponds to an adjective rating between *"OK"* (50.9) and *"Good"* (71.4). During the interviews, we identified several issues that negatively affected the usability assessment. For example, almost all participants found the highlighting of the currently selected sequence path too subtle and therefore had difficulties identifying it. Furthermore, many participants found it hard to understand the relationship between individual simulation sequences and clusters. The behavior when selecting multiple items in the cluster timeline was also considered to be confusing by several subjects. The most commonly requested features were a way to easily mark certain simulation sequences as favorites and the ability to compare them side-by-side. As we had anticipated, many participants did not find the parameter view particularly useful. However, several subjects indicated that the integration of additional interaction functionality such as the filtering of parameter values would make this component more valuable.

In general, the response of the visual effects specialists was particularly encouraging. All of them indicated that they would like to use our system in production as soon as some minor quirks are resolved and some of them were interested in having the current version of the software installed on their workstations immediately. The ability to visually explore parameter variations was highly appreciated, and comments such as *"This can save me many hours of work"* and *"Instead of spending my time doing guesswork, this system does the guesswork for me"* left us with the impression that our approach can provide a valuable addition to the production process. A common use-case the professionals found particularly exciting was the ability to easily generate multiple variations of a particular effect for review by supervisors and clients. Additionally, there were many requests about applying our general concept to other common tasks such as cloth and crowd simulation. Based on the overwhelmingly positive feedback we received, we are currently working on addressing the main usability issues and integrating feature suggestions. Within the upcoming months, we plan to provide an updated version of the software to the visual effects artists for beta testing in production use.

## 8 Conclusion

In this paper, we presented a visualization system for the exploration of simulation parameter settings in visual effects design. Current software tools provide no visual guidance in the parameter selection process and artists have to resort to a time-consuming and cumbersome trial-and-error strategy. Our system samples the parameter space and employs a novel approach for clustering the resulting volumetric time sequences in order to discover characteristic variations in relation to their temporal evolution. Our novel visual representation of the clustering results enables exploration of the simulation space at different temporal levels-of-detail and provides an instant overview. Using our result-driven interactive visual exploration environment gives users the ability to find simulation sequences based on a particular artistic vision.

The main contribution of this paper is the general concept of a visualization system for the phenological exploration of simulation data. To the best of our knowledge, our system is the first that specifically attempts to make modeling of difficult natural phenomena accessible to the non-technical practitioner. Our approach for segmenting and clustering volumetric time series deliberately makes minimal assumptions

and attempts to classify the data based on their observable characteristics. Although our methods were developed with a specific application in mind, the proposed techniques may also be useful in other scenarios. In particular, cloth modeling and other animation effects that are based on intrinsically computationally expensive or mathematically complex models may benefit from the described methods. Furthermore, the investigation of the parameter distribution in relation to clusters solely based on the characteristics of the output data may be an interesting alternative approach to conventional analysis methods for general simulation data. A further direction for future work involves the integration of computational steering into our system. Ideally, one would like to be able to derive the parameters for a set of desired output characteristics from a sparse set of samples. One approach could be to employ key frame information specified using our blending mechanism directly as the basis for an optimization strategy based on a multi-objective evolutionary algorithm [29].

## ACKNOWLEDGMENTS

The authors wish to thank Steve DiPaola and Blair Tennessy for their invaluable help in conducting the user study as well as Melanie Tory for her assistance in designing the protocol. The work presented in the paper was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC).

## REFERENCES

[1] W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski. Visualizing time-oriented data - a systematic view. *Computers & Graphics*, 31(3):401–409, 2007.

[2] H. Akiba and K.-L. Ma. A tri-space visualization interface for analyzing time-varying multivariate volume data. In *Proceedings of EuroVis 2007*, pages 115–122, 2007.

[3] H. Akiba, K.-L. Ma, and N. Fout. Simultaneous classification of time-varying volume data based on the time histogram. In *Proceedings of EuroVis 2006*, pages 1–8, 2006.

[4] H. Akiba, C. Wang, and K.-L. Ma. AniViz: A template-based animation tool for volume visualization. *IEEE Computer Graphics and Applications (to appear)*, 2010.

[5] N. Andrienko, G. Andrienko, and P. Gatalsky. Exploratory spatio-temporal visualization: an analytical review. *Journal of Visual Languages & Computing*, 14(6):503–541, 2003.

[6] M. Ankerst, S. Berchtold, and D. Keim. Similarity clustering of dimensions for an enhanced visualization of multidimensional data. *Proceedings of IEEE InfoVis 1998*, pages 52–60, 1998.

[7] A. Bangor, P. Kortum, and J. Miller. An empirical evaluation of the system usability scale. *International Journal of Human-Computer Interaction*, 24(6):574–594, 2008.

[8] A. Bangor, P. Kortum, and J. Miller. Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of Usability Studies*, 4(3):114–123, 2009.

[9] D. Birant and A. Kut. ST-DBSCAN: An algorithm for clustering spatial-temporal data. *Data & Knowledge Engineering*, 60(1):208–221, 2007.

[10] G. Daniel and M. Chen. Video visualization. In *Proceedings of IEEE Visualization 2003*, pages 409–416, 2003.

[11] N. Elmqvist, J. Stasko, and P. Tsigas. Datameadow: A visual canvas for analysis of large-scale multivariate data. In *Proceedings of VAST 2007*, pages 187–194, 2007.

[12] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of Knowledge Discovery and Data Mining 1996*, pages 226–231, 1996.

[13] J. Fails, A. Karlson, L. Shahamat, and B. Shneiderman. A visual interface for multivariate temporal data: Finding patterns of events across multiple histories. In *Proceedings of VAST 2006*, pages 167–174, 2006.

[14] Z. Fang, T. Möller, G. Hamarneh, and A. Celler. Visualization and exploration of time-varying medical image data sets. In *Proceedings of Graphics Interface 2007*, pages 281–288, 2007.

[15] A. Hanjalic. Shot-boundary detection: unraveled and resolved? *IEEE Transactions on Circuits and Systems for Video Technology*, 12(2):90–105, 2002.

[16] A. Hanjalic and H. Zhang. An integrated scheme for automated video abstraction based on unsupervised cluster-validity analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1280–1289, 1999.

[17] A. Hanson and P. Heng. Four-dimensional views of 3D scalar fields. In *Proceedings of IEEE Visualization 1992*, pages 84–91, 1992.

[18] M. Harrower and C. Brewer. ColorBrewer.org: An online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003.

[19] S. Havre, E. Hetzler, P. Whitney, and L. Nowell. ThemeRiver: visualizing thematic changes in large document collections. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):9–20, 2002.

[20] H. Hochheiser and B. Shneiderman. Dynamic query tools for time series data sets: Timebox widgets for interactive exploration. *Information Visualization*, 3(1):1–18, 2004.

[21] S. Johansson and J. Johansson. Interactive dimensionality reduction through user-defined combinations of quality metrics. *IEEE Transactions on Visualization and Computer Graphics*, pages 993–1000, 2009.

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

[23] K.-L. Ma. Image graphs - a novel approach to visual data exploration. *Proceedings of IEEE Visualization 1999*, pages 81–513, 1999.

[24] K.-L. Ma. Visualizing time-varying volume data. *Computing in Science & Engineering*, 5(2):34–42, 2003.

[25] J. Marks, B. Andalman, P. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of ACM SIGGRAPH 1997*, pages 389–400, 1997.

[26] M. Monks, B. Oh, and J. Dorsey. Audioptimization: Goal-based acoustic design. *IEEE Computer Graphics and Applications*, 20(3):76–91, 2000.

[27] P. Muigg, J. Kehrer, S. Oeltze, H. Piringer, H. Doleisch, B. Preim, and H. Hauser. A four-level focus+context approach to interactive visual analysis of temporal features in large scientific data. *Computer Graphics Forum*, 27(3):775–782, 2008.

[28] C.-W. Ngo, T.-C. Pong, and H.-J. Zhang. On clustering and retrieval of video shots. In *Proceedings of Multimedia 2001*, pages 51–60, 2001.

[29] S. Obayashi. *Evolutionary Multi-Objective Optimization and Visualization*, chapter 16, pages 175–185. Springer, 2005.

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

[31] R. Smith, R. Pawlicki, I. Kókai, J. Finger, and T. Vetter. Navigating in a shape space of registered models. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1552–1559, 2007.

[32] A. Treuille, A. Lewis, and Z. Popović. Model reduction for real-time fluids. *ACM Transactions on Graphics*, 25(3):826–834, 2006.

[33] B. Truong and S. Venkatesh. Video abstraction: A systematic review and classification. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 3(1):1–37, 2007.

[34] J. van Wijk and R. van Liere. Hyperslice: visualization of scalar functions of many variables. In *Proceedings of IEEE Visualization 1993*, pages 119–125, 1993.

[35] L. Wang, Y. Zhang, and J. Feng. On the Euclidean distance of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1334–1339, 2005.

[36] M. Wohlfart and H. Hauser. Story telling for presentation in volume visualization. In *Proceedings of EuroVis 2007*, pages 91–98, 2007.

[37] J. Woodring and H.-W. Shen. Chronovolumes: a direct rendering technique for visualizing time-varying data. In *Proceedings of Volume Graphics 2003*, pages 27–34, 2003.

[38] J. Woodring and H.-W. Shen. Multi-variate, time varying, and comparative visualization with contextual cues. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):909–916, 2006.

[39] J. Woodring and H.-W. Shen. Multiscale time activity data exploration via temporal clustering visualization spreadsheet. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):123–137, 2009.

[40] J. Woodring, C. Wang, and H.-W. Shen. High dimensional direct rendering of time-varying volumetric data. In *Proceedings of IEEE Visualization 2003*, pages 417–424, 2003.

[41] E. Wu, Y. Liu, and X. Liu. An improved study of real-time fluid simulation on GPU. *Computer Animation and Virtual Worlds*, 15(34):139–146, 2004.