STAG: Smart Tools and Applications in Graphics (2019)
M. Agus, M. Corsini and R. Pintus (Editors)

Visual Representation of Region Transitions in Multi-dimensional Parameter Spaces

O. Fernandes, S. Frey, G. Reina, and T. Ertl

Visualization Research Center, University of Stuttgart

Abstract

We propose a novel visual representation of transitions between homogeneous regions in multi-dimensional parameter space. While our approach is generally applicable for the analysis of arbitrary continuous parameter spaces, we particularly focus on scientific applications, like physical variables in simulation ensembles. To generate our representation, we use unsupervised learning to cluster the ensemble members according to their mutual similarity. In doing this, clusters are sorted such that similar clusters are located next to each other. We then further partition the clusters into connected regions with respect to their location in parameter space. In the visualization, the resulting regions are represented as glyphs in a matrix, indicating parameter changes which induce a transition to another region. To unambiguously associate a change of data characteristics to a single parameter, we specifically isolate changes by dimension. With this, our representation provides an intuitive visualization of the parameter transitions that influence the outcome of the underlying simulation or measurement. We demonstrate the generality and utility of our approach on diverse types of data, namely simulations from the field of computational fluid dynamics and thermodynamics, as well as an ensemble of raycasting performance data.

CCS Concepts

• ***Human-centered computing*** → ***Visualization techniques;*** • ***Computing methodologies*** → *Simulation tools;* • ***Applied computing*** → *Physical sciences and engineering;*

1. Introduction

The ever-growing computational power allows to use *ensembles* to examine application behavior and how it changes in relation to input (parameter) values. Essentially, an ensemble is a set of *samples* produced by the application with varying input *parameters* from a certain *parameter space*. Depending on the application, the properties of this parameter space can range from a handful of discrete choices to a continuous multi-dimensional space. Typically, for the scientific applications targeted by our visualization, the parameter space is continuous and has multiple dimensions of interest, with a classical example being numerical simulations; one target case could be a computational fluid dynamics (CFD) simulation with the Reynolds number as a parameter. We denote the result of a simulation, or any process operating on the inputs, as *outputs*. For the CFD example these would be time steps containing state variables of the simulation, such as velocity and pressure.

Gaining an understanding of a simulation's behavior when varying input parameters is not an easy task. Manually examining the results becomes tedious, if not impossible, as the ensemble's sample number grows. While aggregation of similar samples simplifies getting an overview of the data, the location of members in parameter space is lost. In addition, how changing a parameter would affect the result is difficult to derive from a simple aggregation. In many cases, the domain scientist is also interested in the inverse analysis, i.e. having clusters of similar results, determine which change of parameters was responsible for the change in simulation behavior.

In this work, we introduce a visual representation depicting which parameter changes lead to transitions between different sample clusters with similar output, and also show the varying impact of parameters in different regions of the parameter space. To this end, we partition the parameter space into regions of similar behavior. We enable the examination of parameter changes by encoding the transitions between these regions into a glyph, allowing for identification of the effects of parameter changes as well as elucidation of the correlation of parameters. Our approach can also assist with finding the cause for outliers within the ensemble, as well as yield information on the impact of parameters at a glance. In addition, the customized glyph allows for locating a region in multi-dimensional parameter space, and give an impression of its size.

The paper is structured as follows. First, we give an overview of related work (section 2). In section 3, we motivate design choices, and give an overview of the individual steps of our approach. On this basis, we discuss our visual representation for analyzing region transitions (section 4). In section 5, we provide technical details for clustering similar ensemble members using Self-Organizing Maps. We present case studies for different applications in section 6, and

discuss limitations and directions for future work in section 7. We conclude our paper in section 8.

2. Related Work

As predicted by Obermaier and Joy in their 2014 article [OJ14], the increasing computational resources advance ensemble generation in general, and consequently require the development of adequate visualization approaches. A comprehensive survey dealing with parameter space exploration in general was conducted by Sedlmair et al. [SHBP]. Recent challenges of ensemble/parameter space visualization were evaluated at the example of material fracturing data by Crossno [Cro18]. Especially the analysis of ensembles on the basis of graphs are a popular approach in this context, as discussed in respective recent surveys [WT, WHLS18].

Similar to our approach, Bruckner and Möller [BM10] initially cluster ensemble members. However, their focus differs in that they aim to identify an input parameter set based solely on the analysis of the output (in their case, renderings of smoke simulations). Instead, our approach aims to not only achieve a categorization of the outputs, but especially to identify structures in the *input* parameters, with a focus on isolating and relating certain simulation behavior to specific parameter changes, and to present them in a global context of the parameter space's structure. Parameter optimization, but with a focus on image analysis, was done by Pretorius et al. [PBCR11], and for procedural geometry creation by Beham et al. [BHGK14]. The Paraglide system by Bergner et al. [BSM*13] implements several analysis methods to examine generic parameter spaces. Gerber et al. [GBPW10] combine topological and geometric techniques for the interactive visualization of high-dimensional scalar fields.

Streamlines based on CFD ensembles were examined by Ferstl et al. [FBW16] using principal component analysis to reduce the dimensionality. Employing the use of "limiting" lines and surfaces, the general behavior of an ensemble set can be extracted. The same group went on to develop a complementary comparative visualization using a time-hierarchical clustering to further reduce the representation [FKRW17]. Another dimensionality-reducing strategy was employed by Bergner et al. [BPFG11]. Their work consists of an interactive approach to enable a continuous analysis of a sampled parameter space with respect to multiple target values, employing statistical methods and uncertainty visualizations to guide users to interesting phenomena. A more general approach to dimensionality reduction is presented by Ingram et al. [IMI*10], which abstracts the interactive analysis into defining operators and expressions on the data. A visualization for vector field ensembles based on comparing pathlines was proposed by Liu et al. [LGZY16] offering some robustness to missing data or varying sampling rates. Hao et al. [HHB16] focus on ensembles where members have a time correlation, offering a wide selection of methods for partitioning the ensemble set. Ensemble members may also be related but not directly comparable. Wang et al. [WLSL17] address this in the context of data differing in resolution with a customized parallel coordinate method to correlate ensemble members. Whitaker et al. [WMK13] extract statistical data for contours in climate-related data, and extend a box plot-based visualization to enable the exploration of weather phenomena and other CFD data.

Interactive approaches include abstractions of ensemble data via different representations, which can then be examined using a visual analytics tool. Shu et al. [SGL*16] use a combination of representing the 2D parameter space as a map, bar charts and star glyphs to visualize characteristics of subsets in the ensemble. For a multi-stage simulation process, Splechtna et al. [SMG*15] developed an approach enabling analysts to easily employ various automatic methods to explore and classify the heterogenous parameter spaces. Another visualization using comparisons of classifications was developed by Unger and Schumann [US09]. Here, the state of a simulation is classified and processes connecting these states are visualized as a graph. Our method is related to this line of work as it is based on a clustering of ensemble members. However, we do not focus on the clustering scheme itself, but focus on the *relations* between clusters. More specifically, many methods produce a visualization that may by integrated into our approach as representations of a specific cluster or unit (cf. section 4). In reverse, our visual representation could be integrated in visual analytics approaches for ensemble analysis to give an overview on parameter space characteristics, and provide a starting point to explore the impact of parameter changes.

We use unsupervised learning, more concretely Self-Organizing Maps (SOMs) first presented by Kohonen et al. [Koh98], as a part of our approach. In general, learning-based methods have great potential in data analysis in general and visualization in particular (e.g. [Ma07]) For instance, for explaining features inside multivariate, volumetric data, Fuchs et al. [FWG09] adapted an evolutionary search algorithm to assist a user in the formalization of respective hypotheses. Most prominently, machine learning has widely been adopted in the area of visual analytics, and we would mainly refer the reader to the recent state-of-the-art report by Endert et al. [ERT*17] for a comprehensive overview. Depending on the use case, SOM visualizations typically present the information of mean cell vectors, or use concrete data items for cell representations (e.g., [BWK*13, Ves99]). For instance, Andrienko et al [AAB*10] investigate how SOMs are integrated into the visual analysis process of spatio-temporal data. They integrate a SOM matrix where the user can interactively modify the parameters and observe the changes in the results in different visual representations, e.g. where space is represented in time, and the time is represented in space. Recently, Sacha et al. [SKB*18] presented a multi-stage visual analytics approach for iterative cluster refinement that uses SOMs to analyze time series data, offering the analyst a visual platform to analyze intermediate results.

3. Motivation, Design Choices, and Overview

Our technique works for ensembles with arbitrary continuous parameter spaces. It is independent from the type of the investigated application, and designed to give an expressive, static overview of parameter space characteristics. The primary goal of our visualization is to relate changes of parameters to respective changes of the outputs in the examined simulations, and vice versa. The basis for this is a partitioning of the parameter space into regions of similar behavior, as has been done in various related parameter space exploration techniques (cf. section 2). In particular, we investigate parameter changes across adjacent regions to help the user understand the impact of parameters and identify thresholds for significant changes in the simulation output. Additionally, our visualization

enables to easily judge the overall impact of a certain parameter, in comparison to other parameters. Conceptually, our approach can serve as an initial step giving an expressive overview on the relation between parameter space and simulation results, and may also be integrated with a variety of solutions from previous work for further exploration.

In fig. 1, we give a schematic overview on the different steps of our approach. Here, we also exemplify it by means of a two-dimensional parameter space $p_1 \times p_2$ with a sample count of 9, starting with a regular sampling in the ranges r_1 and r_2 (fig. 1(a)). All steps work accordingly for higher-dimensional parameter spaces.

3.1. Sorting into Bins by Similarity

On the basis of our sampled parameter space (fig. 1(a)), we evaluate the outputs of each simulation by clustering samples into bins of similar behavior, whereas the bins are implicitly sorted such that similar bins are close. Note that we use the term *units* and *bins* synonymously in the following. As a basis for this clustering, we use pairwise comparisons to determine the similarity of individual members. The actual comparison function is determined by the analysis task at hand, and varies depending on the outputs of the investigated systems (e.g, a field in the case of a CFD simulation). Due to computational constraints, a full pairwise comparison (with $\mathcal{O}(n^2)$ complexity) is not always possible, e.g. 2000 samples with a comparison method taking 0.5 seconds per execution would take over 3 weeks to complete on a single workstation. To address this, we generate a feature vector $\mathbf{f}_s$ for each sample s, using only a subset of samples to compare to. As this is inherently tied to the data set and the analysis task, we defer the detailed explanation of the comparisons and the feature vector generation to the discussion of the respective case study in section 6. Based on the feature vector $\mathbf{f}_s$, we use an unsupervised learning algorithm (1D Self Organizing Map, SOM) to sort and partition the samples into sets of similar behavior (with respect to the application output). The usage of SOMs in this context explained in detail in the implementation section (cf. section 5).

For simplicity, our example feature vector only has a single component in fig. 1(a). The sorting determined by the SOM is denoted by the coloring of the samples into purple, pink and yellow shown in fig. 1(b). Same color means the sorting determined the samples to be in the same unit. So far, this ignores any information about the underlying parameter space, but merely evaluates the respective application results.

3.2. Parameter Space Connectivity

Using the results from the previous step, we now consider the partitioning of the parameter space. Samples that are considered similar do not necessarily have to have any relationship in the parameter space. To enable the examination of this relationship, we divide similar samples into connected components in the parameter space. In fig. 1(c), regions are marked with a border in the color of the unit of the samples they contain, and denoted by arabic numerals in the lower left corner for further reference.

Extracting the Interface Data Before generating the final visualization, we extract what we call *interfaces* from the data structure, shown in fig. 1(d) via dashed silhouettes. We define a *transition* to connect two samples with an axis-aligned edge in the parameter space, which means that exactly one parameter changes between those samples. We only evaluate axis-aligned immediate neighbors for the visualization, as larger parameter changes, or changes across multiple parameters are handled implicitly by the Similarity Binning section 3.1: the binning exposes the transitions with the 'largest dissimilarity' (i.e. comparisons within regions are regarded as uninteresting with respect to the comparison function). These transitions can be thought of as a generalization of the location and direction of the highest gradient in terms of similarity, moving from one region to a neighboring one.

The transitions between neighboring regions form so-called interfaces. These interfaces are depicted in fig. 1(d) with dashed frames and enumerated with roman numerals, and are represented in the final visualization in a stacked fashion. The specifics of this representation are discussed in detail in section 4.

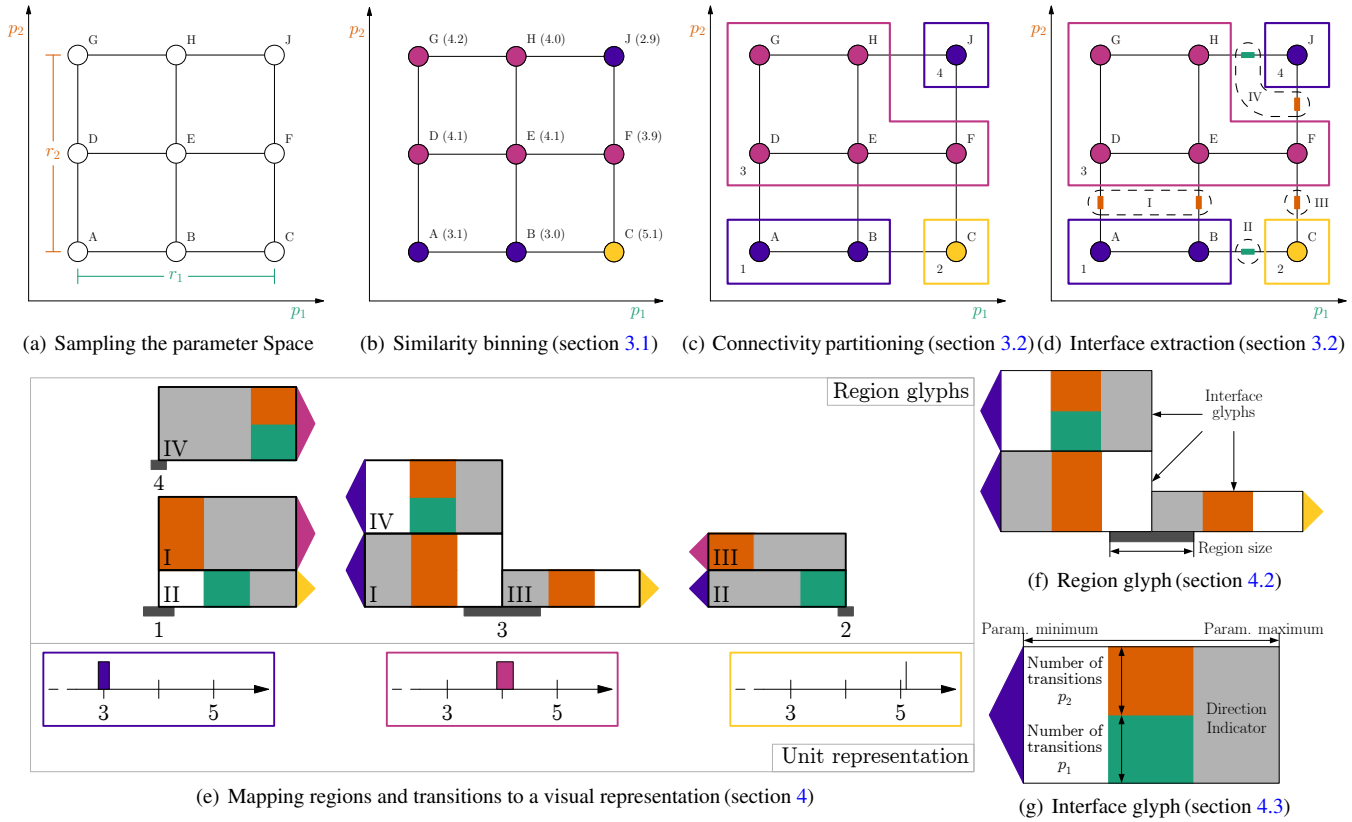
Visualization via Region Glyphs We then proceed to generate the glyphs for each region established in section 3.2 (e.g. fig. 1(f)). The glyphs are arranged in a layout of columns, where each column represents a unit from the Similarity Binning, as can be seen for the demonstration data set in fig. 1(e). Each column has a so-called unit representation on the bottom, giving an impression of the regions contained in that unit. Each region glyph in turn is made up from a set of so-called interface glyphs, containing details of the actual parameter transitions. Our visual representation is explained in detail below.

4. Visual Representation

As introduced above, a region is a set of similar samples which is connected in parameter space. A transition denotes two adjacent samples in parameter space which are not in the same region. In this paper, we restrict ourselves to parameter changes in a single dimension only, while all others remain constant. This allows us to clearly attribute differences between the connected samples to the change of this very parameter (cf. section 7 for a closer discussion of merits and limitations of this design decision). An interface between two regions A, B is a collection of transitions for which one sample lies in A while the other sample belongs to B. In our visualization, we associate each region with a glyph akin to a stacked bar chart.

4.1. Unit Column

The upper part of each unit column shows a region glyph for each region associated with the respective unit (discussed in more detail below in section 4.2). The lower part of each unit column provides a suitable depiction characteristic of the contained ensemble members. Different mappings and metaphors can be chosen here depending on the underlying application and aspects of interest. Note that developing suitable mappings for this is not the primary focus of this paper. For the case studies in this paper having a simple scalar value for each member, we simply present the range of this scalar included in the unit (e.g., cf. the introductory example in fig. 1(e)). For more



(a) Sampling the parameter Space (b) Similarity binning (section 3.1) (c) Connectivity partitioning (section 3.2) (d) Interface extraction (section 3.2)

(e) Mapping regions and transitions to a visual representation (section 4)

(f) Region glyph (section 4.2)

(g) Interface glyph (section 4.3)

Figure 1: *Overview of the individual steps of our approach. (a) Starting from the sampled parameter space, (b) similar samples are clustered first. (c) These clusters are then further decomposed into regions of connected components in parameter space. (d) Next, transitions between individual regions are determined. (e) Finally, regions and interfaces are directly mapped to our visual representation. For illustration purposes, arabic numbers indicate the mapping of regions from (c) to glyphs and roman numerals indicate particular transitions. (f) The Region Glyph is composed from individual interface glyphs, shown in (g).*

complex cases, we use a direct visualization of a representative sample. Many approaches mentioned in the related work section offer more advanced alternatives, e.g. the star glyph used in Bruckner's cluster representation [BM10]. In all cases, the unit representation is marked by a colored frame. The color represents the 1D coordinate as resulting from the Similarity Binning (cf. section 3.1) using a linear transfer function mapping. All regions in one column are contained in the same, respective unit, and accordingly associated with the same color.

4.2. Region Glyph

Each region is represented by a region glyph. The region glyph consists itself of two columns, each consisting of stacks of interface glyphs. The left stack contains all interfaces which connect to a region in a unit column to the left of the current unit, and the right stack contains all interfaces connecting to regions to the right. For the left stack, the left border is marked with several arrows, one for each interface, the so-called unit indicators (and vice versa for the right stack). This unit indicator is filled with the color of the unit that the neighboring region is associated with. For each interface glyph, as well as for the entire region glyph, the height encodes how many individual transitions incur a change of simulation behavior to neighboring regions. In addition, the size of the region can be determined by a small dark gray horizontal indicator below each glyph, in proportion to the samples contained in this region.

4.3. Interface Glyph

The information encoded into the interface glyphs helps investigate which parameter changes are actually responsible for differences in ensemble members. An annotated example is shown in fig. 1(g). For this, the transitions, as extracted in section 3.2, are directly depicted, and within each interface glyph, for each parameter a colored block is drawn, from bottom to top. Note that each interface appears twice in the visualization, once for each of the two regions it connects. The color encodes the dimension in which the change occurred, while discarding the information of other dimensions (which remain constant for any specific transition). While this information might be interesting for further analysis, displaying it for all transitions easily overloads the visualization. The height of the block is directly proportional to the number of transitions associated with this interface and this parameter, and enables a relative comparison of the transition count. In other words, it depicts how large the multi-

dimensional "surface" is that separates the regions in parameter space. The horizontal dimension of a block is used to map the range of the transitioning parameter, normalized to their respective range. Each block covers the parameter range from the left end (minimum) to the right end (maximum) as annotated in fig. 1(g).

While the visualization shows at which values of a parameter transitions occurred and how many there are, the direction of change is shown encoded as gray areas to the left and right of each block. While a gray area to the *right* indicates an *increase* of the parameter to effect the transition, a gray area to the *left* indicates a *decrease*. Conversely, a white area to the right means that, for this particular interface, increasing the parameter will never lead to results that lie in the neighboring region (same for the left and decreasing). This means that a parameter change corresponding to a white area can only generate a transition through a different interface. The height of the gray areas on the right is proportional to the number of transitions for which the parameter value is increased to achieve a change of behavior (same for the left area and transitions where the parameter is decreased). We observed that in most cases, a parameter only changes in one direction for a given interface.

The two interface glyphs representing a connection are essentially mirrored. However, as only the start of the transition (i.e. the parameter value for the sample within the region) is taken into account when rendering the colored block, a discrepancy of one sampling interval in size is visible. While mostly irrelevant for denser samplings, this becomes apparent for coarse sampling rates.

If the parameter space decomposes into regions with small interfaces, examination of these might become difficult, as region and interface glyph heights go below pixel size. To address this issue, the respective glyph may be simply scaled vertically.

Visualizing transitions as blocks assists in quickly judging how impactful a parameter is when moving through parameter space to a region exhibiting different behavior: while the vertical extension denotes the impact of the parameter (i.e the number of transitions associated with this dimension), the horizontal axis shows at which values the transitions occur. The saturation encodes the distribution of transitions associated with this parameter along the horizontal axis. For example, if most transitions occur at the maximum value, the right end of the block will be more saturated; if transitions occur across all values uniformly, the block will be evenly colored.

5. Similarity Binning via Self-Organizing Maps

For our implementation of the Similarity Binning (i.e, sorting ensembles with respect to the similarity of ensemble members), we use a Self-Organizing Map (SOM). SOMs are based on feature vectors $\mathbf{f}_s$, defined for all samples s of an ensemble, which provide a simple abstraction of a sample's output. Determining these features is strongly tied to both the data set and and the analysis task, and details on the computation of the feature vector in this work are given alongside the respective case study in section 6.

SOMs are an unsupervised learning approach to produce a low-dimensional, discretized representation of a set of high-dimensional features. The learning process performed by the SOM algorithm associates each ensemble sample s with one of the vectors $\mathbf{u}_i \in U$, called unit. Here, samples associated to a certain unit u_i are considered similar. This association is determined on the basis of the Euclidean distance of the feature vector $\mathbf{f}_s$ to the vector u_i. The number $n = |U|$ of units created is fixed a priori, and decided by the user.

In addition, SOMs not only yield an evaluation for each of the ensemble samples into a disjoint partitioning, but also establish an overall similarity relation between the u_i. This relation is expressed as an ordering of the individual units, with samples associated with one unit being similar to samples in its predecessor and successor (as opposed to units further away). In other words, the ordering of the $\mathbf{u}_i$ derived from a SOM training reflects a gradual change of the sets of samples s associated with each $\mathbf{u}_i$. We utilize this ordering in the layout of the nodes in the final visualization.

Our learning approach establishes the final values for the units' vectors $\mathbf{u}_i$. For each element $\mathbf{u}_i \in U$ we randomly pick one of our feature vectors $\mathbf{f}_s$, and assign the value of the feature vector to the element $\mathbf{u}_i$ (i.e., $\mathbf{u}_i \leftarrow \mathbf{f}_s$). We then start the training, utilizing competitive learning in multiple passes.

In each pass, we loop over all input feature vectors $\mathbf{f}_s$ of all samples s in random order. For each feature vector $\mathbf{f}_s$, we pick the unit $\mathbf{u}_i$ with the smallest Euclidean distance. We perform an update of this $\mathbf{u}_i$ by "pulling" the vector associated with $\mathbf{u}_i$ toward $\mathbf{f}_s$ using a weight w, i.e. $\mathbf{u}_i \leftarrow (1-w)\mathbf{u}_i + w\mathbf{f}_s$. However, not only $\mathbf{u}_i$ is updated, but also the units within a vicinity v, thus changing the values for $\mathbf{u}_j, j \in [i-v, i+v]$. The weight of the update w linearly decreases with the distance $|i-j|$ to the unit $\mathbf{u}_i$, with the radius of considered units initially being half the number of total units in our implementation. This process is repeated each pass, while the considered neighborhood v decreases with an increasing number of passes (it is multiplied with a factor of 0.98 after each pass). The learning process is considered to be converged when the changes to $\mathbf{u}_i$ are negligible.

Table 1: *Sampling grids for the parameter spaces in the case studies. The number in parentheses indicates the total number of samples. For Water Model, values are percentages. Absolute values can be extracted from the official IAPWS Formulation [IAP18]. The units for Kármán Vortex Street are in units of the solver characteristical length. More details are given with the specific case study.*

	Voronoi (16384)		Performance (36864)			
	x	y	d	ϕ	s	v
Min	0	0	0.6	0°	0.25	600
Max	128	128	2.0	360°	2.00	2000
Step	1	1	0.2	5°	0.25	200
	Kármán (1287)			Water Model (275)		
	R	o	r	Q	σ	ε
Min	40	0.0	0.3	50	90	80
Max	100	0.4	0.8	150	110	120
Step	5	0.05	0.05	10	5	10

6. Case Studies

We now apply our visual representation to analyze various ensembles. First, we showcase how information is prepared using a simple, synthetic data set (section 6.1). We continue with data from different domains: an ensemble of fluid dynamics simulations (section 6.2) as well as an ensemble generated from a thermodynamics simulation (section 6.3). To demonstrate the generality of the approach, we finally analyze an ensemble of measured performance data from volume raycasting with varying rendering parameters (section 6.4).

6.1. Case Study 1: Voronoi Image

The first data set is a synthetic data set of a square image decomposed into distinct areas. Specifically, some random points were chosen and associated Voronoi cells were generated. Each cell is then colored black, gray or white (cf. fig. 2(a)). For this data set, the feature vector is kept as simple as possible to allow for easy tracking of data flow and analysis of algorithmic settings. This data set can be interpreted as a "simulation" taking 2 input parameters (x, y - coordinates) and producing a color as result:

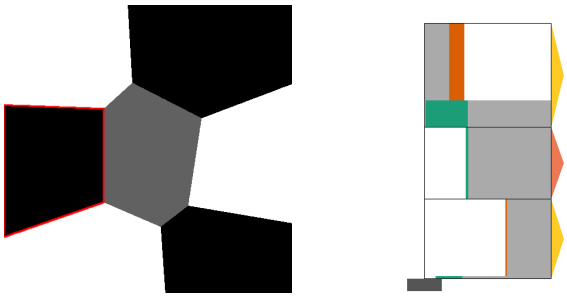
Image X Axis x **(Green).** Vary the value of the x coordinate.
Image Y Axis y **(Orange).** Vary the value of the y coordinate.

Using the x, y coordinates as parameter space, we just use the (grayscale) color of the pixel at x, y as a (single-component) feature vector.

Using a SOM with $n = 6$ units, the data gets sorted into three categories, as expected, while the other units remain empty (units renamed as follows: u_0=black, u_1=gray, u_2=white, while keeping the SOM-determined ordering). Applying the final partitioning using the neighborhood in parameter space, we obtain a set of 7 region components, and our visualization approach can be seen in fig. 2(c). On the horizontal axis, the identified clusters are listed in the order determined by the SOM. The unit representation below each column consists of the original pixel color.

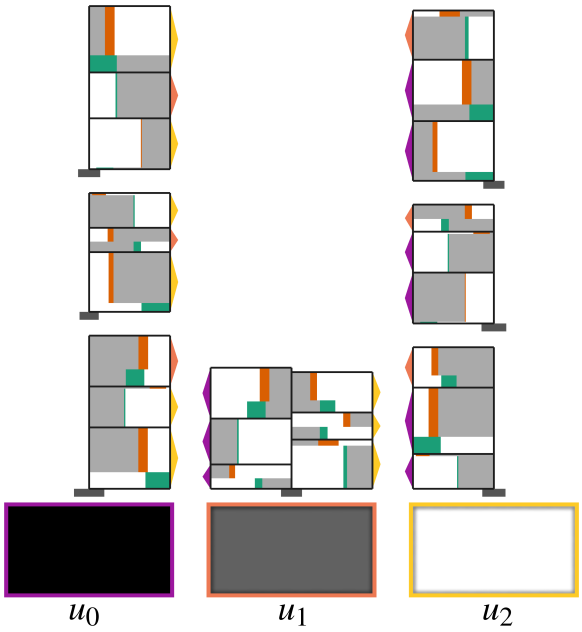
First, the category of each region can be easily compared to surrounding regions. For example, the center glyph, representing the gray region, has connections to both lighter (white) and darker (black) regions, as there are stacks on both the right and the left side of the center. Also the distance in terms of the category of individual interfaces can be directly extracted from the unit indicators. For example, consistent with the actual image, the black region represented in the top left glyph (associated with the marked region in fig. 2(a), and shown enlarged in fig. 2(b)) shows two connections to the yellow category (white regions), and one to the orange one (gray region).

To find out how to get from regions yielding black to a region yielding white (yellow category), the colored blocks within the individual stacks need to be examined (fig. 2(b)). A colored block is drawn for each parameter dimension with at least one transition, and the height of this block encodes the number of transitions associated with this dimension. One can see from the middle transition in the glyph, connecting to a region in u_1, that this interface is dominated by the green parameter (x): there is only a green block and, since the region border is perpendicular to the parameter, only one value generates a transition, resulting in a vertical green line. The gray block to the *right* of the green line implies that x needs to be *increased*.



(a) The Voronoi image used as synthetic parameter space set.

(b) Closeup of the black region marked in (a).



(c) Analysis of the Voronoi Image dataset.

Figure 2: *Synthetic data set* ***Voronoi Image****, with source data set and our approach.*

Also, given that all blocks in all glyphs are evenly colored, the transitions are evenly distributed among the respective parameter range. While these deductions can be made easily directly from the simple data set fig. 2(a), this is increasingly difficult with higher number of dimensions of the parameter space.

6.2. Case Study 2: Kármán Vortex street

For this ensemble data set, a Kármán vortex street fluid dynamics simulation was executed for varying parameters. The simulation code used is the OpenLB Lattice-Boltzmann solver [KTHRT*16]. Three different parameters were modulated for the ensemble, the details of the sampling are shown in table 1:

Reynolds Number R **(Green).** The *Reynolds* number is a dimensionless measure in CFD simulations which combines lengths and viscosity to easily compare differently scaled setups.
Offset o **(Orange).** The *Offset* parameter defines the vertical position of the circular obstacle in cross flow section.

Radius r (Blue). The *Radius* parameter defines the radius of the circular obstacle.

The analysis task for this ensemble is to explore the overall behavior of the simulation with a focus on turbulent phenomena, i.e. to partition the parameter space into turbulent and laminar behavior, and to determine which parameters cause a transition between these behaviors in different regions of parameter space. This example demonstrates that our approach is also applicable for non-ordinal data, for which we need to rely on pairwise comparison.

Feature vector for Similarity Binning. We first define a comparison function $d(\cdot,\cdot)$ for two individual samples s_i and s_j, which represents the difference in velocity magnitude per voxel (root mean square error) in a given time step of the simulation. This reduces the comparison between two samples to a scalar, the so-called similarity distance d, with a small value meaning more similar. To avoid the huge impact the geometry of the obstacle itself would have on such a comparison, we only use the right half of the simulation's spatial domain. This does not contain the obstacle, and the flow's velocity magnitude itself dominates the comparison. We want our comparison to consider two simulations similar if the distances between vortices are similar, as this is an indicator for how turbulent the vortex street is. However, since a similar vortex pattern may still be time-shifted between two simulations, a direct comparison between the same time steps in both simulations is not sufficient. Hence, we perform a pairwise comparison for each of the last 16 time steps of s_i with each of the last 16 of s_j, instead of just a fixed time step. 16 time steps are sufficient to cover at least one whole period of the vortex shedding pattern for any of the input parameters, after which the pattern repeats itself. The closest match of these comparisons is then used as the comparison value for simulations s_i and s_j.

Now that we have a pairwise comparison, we can define a feature vector. First, we want to select a set of *reference* samples W from the ensemble. These samples should ideally cover all features that we want to detect for our analysis task. Conversely, this means the samples in the set should be as diverse (as in "not similar") as possible. We choose these samples using the following method.

We start with a reference set W containing one random sample w_0, $W = \{w_0\}$. Now, further references w_i are added iteratively: For each sample s of the ensemble the *minimum distance* d_{min} to W is calculated. To determine d_{min} for a sample s, a comparison is performed against all reference samples w already contained in W. The shortest distance of a sample s to any w, i.e. the distance of s to the most similar w, is now defined as d_{min}. In this fashion, a d_{min} is now calculated for each sample s. The s with the *largest* d_{min} is selected as the next w_i, added to W, and the next iteration is started. The iteration stops at a user-defined size of W, but can be also be run until the largest d_{min} is below a certain threshold.

In addition, this method does not have to start with an empty set for W. A domain scientist could manually pick a set of samples to initialize W with. Note that this method only needs comparisons of the order $\mathcal{O}(n)$, and can be progressively enhanced.

For the Kármán ensemble, we initialized the reference set W with the samples simulated for the corners of the parameter space, and added an additional 12 samples w using the above method. For each sample s of the ensemble, we now associate a feature vector with $m = |W|$ components. By setting the i-th component of the vector to the similarity distance d between s the i-th element of W, we generate a vector describing an ensemble sample's relation to the reference set. On this feature vector, the usual Similarity Binning via SOMs (cf. section 3.1, section 5) is performed, and finally our visual representation is computed.

Visual Representation and Analysis. For the unit representation, we show an image of the (normalized) velocity magnitude for the sample with the shortest Euclidean distance to the feature vector of the respective SOM unit u_i (fig. 3).

It can be immediately inferred from the amount of regions in each SOM unit u_i that the detected features and the parameters generating them correlate quite well, as most u_i do not decompose into multiple regions (the small regions in u_1 are artifacts of the clustering and the discretization). With the comparison function operating on the right half of the simulation's spatial domain, one can assume that simulations contained in a region exhibit similar behavior regarding the flow, but do not necessarily have the same obstacle configuration.

The right half (u_3 to u_5) shows that the flow behavior, a per the comparison function, is dominated by the change in obstacle geometry, despite the area containing the geometry itself being excluded from the comparison. This can be seen in the mostly blue and orange colors within the interfaces, encoding *Offset* and *Radius* of the obstacle. In other words, one "extremum" discerned by the machine learning algorithm is mostly due to the obstacle being offset from the center axis of the simulation.

By only examining the unit representation below the columns, one could assume that while samples in u_1 and u_5 are separated, they share some relationship in parameter space, as both exhibit turbulence. In our visualization, the interfaces for these regions confirm that the samples in u_5 have connectivity in parameter space, as they connect to purple regions in u_0 and u_1, as can be seen by the border coloring. A closer examination reveals that parameter changes between these stem mostly from blue and green (*Radius* and *Reynolds*) transitions (e.g. u_4, the topmost interface on the left side). Examining the transitions in the left half (u_0, u_1), the more prominent green transition color reveals that these regions connect to others in the right half and also to each other due to changes in the *Reynolds* number for most values, implying that this parameter dominates the simulation in this region of parameter space.

6.3. Case Study 3: Water Model

The ensemble generated with this simulation consists of testing the macroscopic behavior of water molecules when their microscopic physical quantities are changed. The simulations were generated with the ms2 Molecular Simulator [RKGC*17]. More details can be extracted from the associated paper [HMH*12].

The parameter space for the simulation are physical constants used in the potential for each water molecule. The simulation approximates interactions between water molecules using a Lennard-Jones potential. For details of the parameter space sampling generated refer to table 1. For all parameters, a base value for each physical constant was assumed, and then varied to generate multiple samples.

Charge Q (Green). The electrical charge of the Oxygen atom. This is the key parameter for electrical interactions.

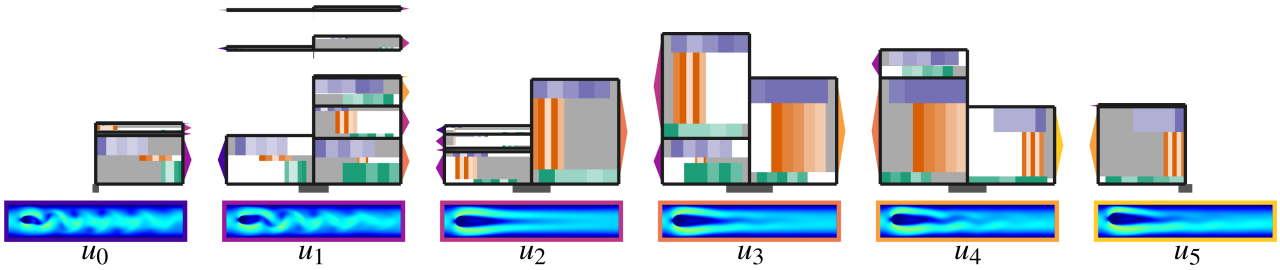


Figure 3: *Kármán Vortex Street ensemble overview. From left to right, one can see adjacent parameter space regions turning from turbulent to laminar (u_0 to u_2), until they get turbulent again (towards u_5). The transition colors shown in the interfaces give a quick assessment for the reason: first lowering the Reynolds number is responsible (mainly green transitions), while the right half of transitions imply that the Offset of the obstacle from the symmetry axis causes the turbulence (orange).*

Epsilon ε (Blue). The "depth" of the well in the Lennard-Jones potential. This mostly affects close-range interaction of molecules.

Sigma σ (Orange). The overall scaling of the Lennard-Jones potential. This can be thought of as the "size" of the molecule.

For each input parameter set (Q, ε, σ), 12 simulations are performed to cover a range of densities and temperatures. The result obtained for each ensemble member are thus 12 value pairs for the macroscopic quantities *inner energy* and *pressure*. These values are compared to experimentally measured values for real water at the 12 given densities and temperatures, to determine how well the simulation approximates the experiment. This comparison is again expressed as the root mean square error. By weighting the error for energy and pressure with a factor provided by the domain expert, the evaluation can be reduced to a single scalar representing the "error" of a simulation associated with a parameter set.

The analysis employing our visualization method is shown in fig. 4. The actual "error" values of the samples contained in the unit are displayed as the unit representation below each column. The unit representation frame color encodes how close the samples contained in the region are to the experiment, from best (purple) to worst (yellow). From the overview, the visualization gives a strong indication that the σ parameter dominates the behavior of the simulation, since many of the transitions are of orange color, especially when you consider that this is the dimension with the coarsest sampling. This means that changing the overall scaling of the molecule has the largest impact on macroscopic properties of the thermodynamical simulation.

Examining the glyph for the "best" region (leftmost, purple), one can make a few observations regarding the ensemble. First, note that transitions tend to be start mostly in the (horizontal) center of each interface glyph, leaving toward both a lower value (gray box to the left) or a higher one (gray box to the right). This implies that simulations closer to base values for Q, σ and ε, which are at the exact center of the column, are more likely to produce better approximations of the experiment. Similarly, examining the worst quality region (rightmost, yellow), the glyph shows that simulations with a high σ value (transitions for two fairly large interfaces are all orange, and located on far right of the interface glyph) can be improved by decreasing σ (gray boxes to left).

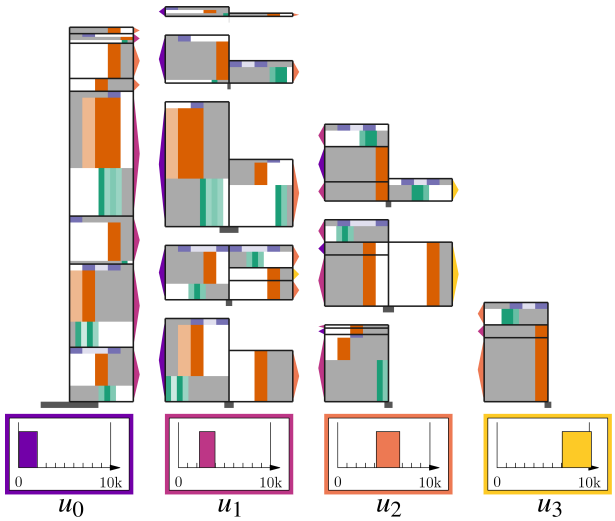


Figure 4: *Water model ensemble overview. Even though having a coarse sampling, σ (orange color) can be easily identified as the most impactful parameter. Also high values for σ generate the largest errors, as can be seen in the highest error region (yellow)*

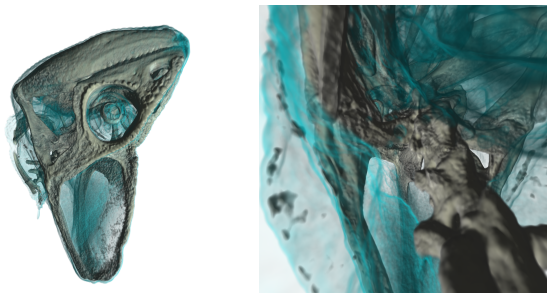
6.4. Case Study 4: Performance Data

Next, we aim to explore the performance characteristics of GPU volume raycasting, one of the fundamental approaches in scientific visualization. Volume visualization is a computationally demanding application, especially for high resolution datasets, which are common nowadays due to advancements in sensor technology and simulation capacities. Therefore, evaluating performance in volume visualization applications and understanding respective performance characteristics is an essential part in this field of research [BMFE19].

For this, a visualization researcher obtained performance data with the following four different parameters:

Ray Step Size (Purple). The sampling distance along a ray consisting of eight values ranging from 0.25 to 2 in terms the smallest side length of a voxel length.

Viewport (Pink). Resolution/number of rays traced through the volume. Here, eight different resolutions were measured, ranging between 600×600 and 2000×2000.



(a) Complete view (b) Close to a bone with large performance impact

Figure 5: *Individual frames of the camera path of the Chameleon data set (*$1024^2 \times 512$*).*

Camera Distance (Green). The camera distance varied in eight steps, ranging from a closeup to an overview.

Camera Rotation (Orange). The camera rotated in an orbit around in the volume in 72 steps (i.e., steps of five degrees).

The measurements were taken from a volume raycaster employing early ray termination and empty space skipping implemented in OpenCL and executed on an NVIDIA GTX970 GPU. A CT scan of a chameleon was used as volume data set in the measurements (cf. fig. 5(a) for an exemplary image). We now aim to study the impact that different parameter changes can have on the performance.

The overall structure of our representation, shown in fig. 6(a), provides insights regarding the general characteristics of the performance data. The units are sorted by decreasing rendering time from left to right. The fastest rendering times are all in one region (yellow unit), meaning that they are all connected in parameter space. Interestingly, essentially all possible transitions in a single dimension in parameter space are present. This means that nearly all individual parameter changes cause a transition to the next slower region. This is due to the fact that for any given parameter in one dimension, there exists another parameter configuration that results in a slower render time. In particular, a low distance to the volume data makes rendering expensive (screen coverage), and for large distances, a high sampling density in image and object space as well as a viewing angle that minimizes the gain from early ray termination can still lead to high render cost. Apart from the smaller size of the other units, this is also a reason why for slower render times, the parameter space is much more segmented: not only has every parameter a significant performance impact by itself, their non-linear interplay also leads to complex performance patterns in parameter space.

Overall, it can be seen that the parameter space typically behaves very smoothly, as indicated by most parameter changes essentially leading to a neighboring unit. In addition, parameter changes behave consistently in terms of their qualitative impact, i.e., the direction of the parameter change that leads to higher or lower render times is consistent in the interfaces of all glyphs. A noteable exception to this is the camera parameter (orange) as can be seen from several glyphs, e.g. in fig. 6(b). The gray blocks for this parameter dimension are found on the left as well as the right side of the orange block. This makes sense, while the others directly steer the sampling density or the image footprint of the volume, the rotation angle has no clear ordering, and advancing on the orbit might lead to an increase or decrease of performance, in particular due the employed acceleration techniques (empty space skipping and early ray termination).

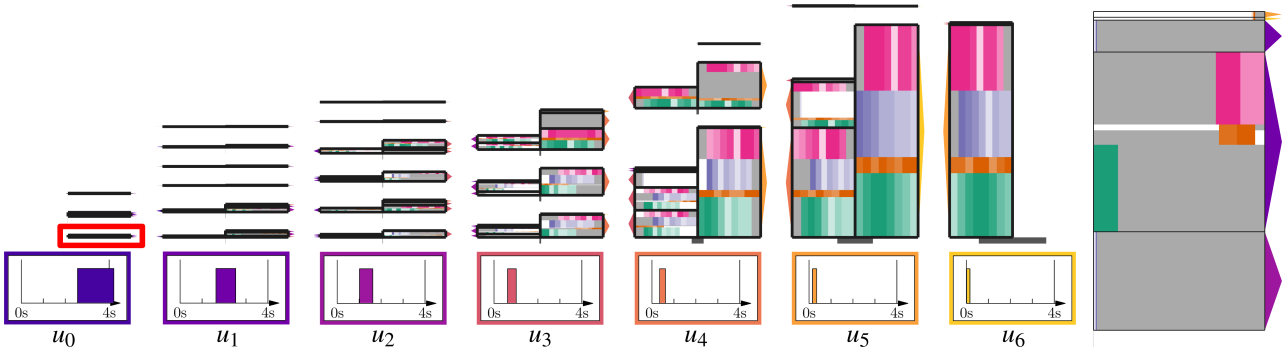
The visualization also shows interesting cases of small parameter changes that have a large performance impact. Visually, this can generally be seen from transition indicators on the glyphs that point to units which are further away. Depending on the users tasks and preferences, such cases can both be identified through user exploration or automatically detected and highlighted algorithmically (e.g., whenever transitions connect fundamentally different regions).

In this example, a closer investigation of the low performance columns on the left (u_0, u_1, purple colors) reveals such a case (fig. 6(b)). The yellow unit transition indicators depict that a single parameter change boosted rendering performance considerably. This phenomenon can be mostly observed when changing the camera rotation, which intuitively should not have such a significant impact on performance. By looking at the output (volume renderings) for the samples at the ends of the respective transitions, we can infer what change in rendering this relates to (fig. 5(b)). The camera is close to a bone, which is essentially dived into with the next step along the camera orbit (orange transitions at the segment with the yellow connection). When this happens, the rendering is cheap, because the bone has high opacity and essentially the computation of all rays almost instantly aborts, due to early ray termination.

7. Timings and Limitations

Timings. While the simulations for the Kármán Vortex Street and Water Model ensemble were performed with in-house hardware, exact timings are not included, since they are unrelated to our approach. Computation times for the simulation step range from a few days to several weeks, using 2 to 6 workstations with varying consumer-grade components (i5-6500 with 4 cores to i7-2600 with 4+4(HT) cores, 8 to 16GB memory), severely limiting the number of samples which can be computed in a reasonable time. The processing steps that are part of our technique were performed on a i7-2600 4+4(HT) core machine with 16GB of memory. The current prototypical implementation of the approach is based on Python unless mentioned otherwise.

The Similarity Binning via SOMs is dependent on the feature vector size and the number of ensemble members. However, the underlying C++ implementation requires a maximum runtime shorter than $t_{SC} < 0.5s$, and is negligible even for larger feature vectors (tested with up 20 components on the Kármán data set (~1300 samples), and with 1 component on the Performance data (~37000 samples)). For the Connectivity Partitioning module, we used a straightforward implementation. Based on adjacency lists, it is applicable for arbitrarily dimensioned, irregular grids, and does not explicitly exploit the regularity present in the grids of our case studies. Hence, this could be easily optimized with an implementation exploiting regularity in the data, and specializing in a given dimensionality of the data grid. All runtimes for this were in the order of a few seconds and depend approximately linearly on the number of ensemble samples. The Interface Extraction times depend linearly on the number of edges in E. The times are negligible across all of our case studies ($< 0.01s$).



(a) Performance ensemble overview. (b) red box from (a)

Figure 6: *Performance ensemble analysis overview. All four parameters have an impact on the high performance unit (yellow, u_6), as all colors can be seen as transitions. This is also due to the fact that the value range covered by unit u_6 is fairly small by comparison. The complex interplay between parameters tends to break the parameter space into smaller partitions for longer rendering time samples.*

The only time-consuming step necessary for our visualization is the computation of the pairwise comparison for the data sets, which is needed for generating a feature vector. The most computationally expensive comparison, done for the Kármán ensemble, required 0.5s per sample pair on average. Using 20 reference vectors, the entire computation takes about 4 hours. Note that once a comparison is calculated, it can be reused in subsequent partitioning runs, provided the analysis task stays the same.

Limitations and Future Work. While our technique enables a good overview on which parameter changes affect the respective output in what way, there are still a number of limitations which can be addressed in future work.

Breaking down interfaces into individual transitions tied to a specific dimension provides insights into how a specific parameter influences the simulation. The correlation between parameters, while still retained, is very difficult to interpret. A simple linear correlation for example can be extracted by having evenly distributed transitions for all dimensions participating in an interface, visualized by all blocks being evenly colored. A more complicated interplay between parameters on the other hand may show fairly complex patterns, which are difficult to deduce from our visual representation. An automatic interpretation of common interplay patterns could be substituted and used instead of the currently employed coloring based on transition density.

Another issue is that the exact location in parameter space of the transitions is lost, as the dimensions remaining constant for a transition are discarded. This could be solved by adding the information to each transition stack, which would add an additional vertical line into each transition block, but would make the result considerably more complex to interpret, as well as further limiting the possible number of dimensions. However, most of these issues can be resolved via interaction, for example using focus and context techniques for the detailed examination of each region and its neighborhood, both in the outputs (i.e. similarity between regions as a whole) as well as in parameter space.

8. Conclusion

Is this paper, we described a novel approach for the visualization of transitions between regions in a multi-dimensional parameter space associated with an ensemble of computations. We focus on continuous parameters typically defined by scientific applications, such as physical constants describing a computational model.

For the analysis of ensembles, we employ similarity metrics calculated between neighboring ensemble members in parameter space, whereas the specific comparison function is tailored to ensemble sample type and analysis task. We employ Self-Organizing Maps to partition the parameter space into segments of similar behavior. Gathering samples within these segments, we further partition into connected components we call regions, which are connected by parameter transitions.

To visualize these resulting regions, we represent them as glyphs resembling stacked bar charts. The associated interface indicators in the glyph then reflect which similarity categories are connected. We then depict the parameter changes leading to a change in simulation behavior, and therefore a change in region, with our novel customized glyph. In general, the glyph gives a visualization of the transitions, their respective location in parameter space, and enables a quick assessment of the impact of a given parameter.

After introducing the technique properties on a synthetic data set, we continued to demonstrate the utility of the approach on several case studies typical for the scientific applications targeted by our approach. This included a CFD simulation ensemble and an ensemble generated for a computational model of water, in which varying physical constants were used. We also added a case study for performance data of a fundamental scientific visualization technique to show the generality of our approach.

Acknowledgements

This research was partially funded by the German Bundesministerium für Bildung und Forschung (BMBF) as part of project "TaLPas" (Task-based Load Balancing and Auto-tuning in Particle Simulations), and the German Research Foundation (DFG) within project D01 of SFB 1313.

References

[AAB*10] ANDRIENKO G., ANDRIENKO N., BREMM S., SCHRECK T., VON LANDESBERGER T., BAK P., KEIM D.: Space-in-time and time-in-space self-organizing maps for exploring spatiotemporal patterns. In *Proceedings of the 12th Eurographics / IEEE - VGTC Conference on Visualization* (2010), EuroVis'10, pp. 913–922. URL: http://dx.doi.org/10.1111/j.1467-8659.2009.01664.x, doi:10.1111/j.1467-8659.2009.01664.x. 2

[BHGK14] BEHAM M., HERZNER W., GRÖLLER M. E., KEHRER J.: Cupid: Cluster-based exploration of geometry generators with parallel coordinates and radial trees. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 1693–1702. doi:10.1109/TVCG.2014.2346626. 2

[BM10] BRUCKNER S., MOLLER T.: Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1468–1476. doi:10.1109/TVCG.2010.190. 2, 4

[BMFE19] BRUDER V., MÜLLER C., FREY S., ERTL T.: On evaluating runtime performance of interactive visualizations. *IEEE Transactions on Visualization and Computer Graphics* (February 2019), 1–1. doi:10.1109/TVCG.2019.2898435. 8

[BPFG11] BERGER W., PIRINGER H., FILZMOSER P., GRÖLLER M. E.: Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction. *Computer Graphics Forum 30*, 3 (2011), 911–920. URL: https://www.cg.tuwien.ac.at/research/publications/2011/Berger_2011_UAE/. 2

[BSM*13] BERGNER S., SEDLMAIR M., MOLLER T., ABDOLYOUSEFI S. N., SAAD A.: Paraglide: Interactive parameter space partitioning for computer simulations. *IEEE Transactions on Visualization and Computer Graphics 19*, 9 (2013), 1499–1512. doi:10.1109/TVCG.2013.61. 2

[BWK*13] BERNARD J., WILHELM N., KRÜGER B., MAY T., SCHRECK T., KOHLHAMMER J.: Motionexplorer: Exploratory search in human motion capture data based on hierarchical aggregation. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2257–2266. URL: http://dx.doi.org/10.1109/TVCG.2013.178, doi:10.1109/TVCG.2013.178. 2

[Cro18] CROSSNO P.: Challenges in visual analysis of ensembles. *IEEE Computer Graphics and Applications 38*, 2 (2018), 122–131. doi:10.1109/MCG.2018.021951640. 2

[ERT*17] ENDERT A., RIBARSKY W., TURKAY C., WONG B., NABNEY I., DÍAZ BLANCO I., ROSSI F.: The state of the art in integrating machine learning into visual analytics. doi:10.1111/cgf.13092. 2

[FBW16] FERSTL F., BÜRGER K., WESTERMANN R.: Streamline variability plots for characterizing the uncertainty in vector field ensembles. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 767–776. doi:10.1109/TVCG.2015.2467204. 2

[FKRW17] FERSTL F., KANZLER M., RAUTENHAUS M., WESTERMANN R.: Time-hierarchical clustering and visualization of weather forecast ensembles. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 831–840. doi:10.1109/TVCG.2016.2598868. 2

[FWG09] FUCHS R., WASER J., GROLLER M. E.: Visual human+machine learning. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1327–1334. doi:10.1109/TVCG.2009.199. 2

[GBPW10] GERBER S., BREMER P., PASCUCCI V., WHITAKER R.: Visual exploration of high dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1271–1280. doi:10.1109/TVCG.2010.213. 2

[HHB16] HAO L., HEALEY C. G., BASS S. A.: Effective visualization of temporal ensembles. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 787–796. doi:10.1109/TVCG.2015.2468093. 2

[HMH*12] HUANG Y.-L., MERKER T., HEILIG M., HASSE H., VRABEC J.: Molecular modeling and simulation of vapor–liquid equilibria of ethylene oxide, ethylene glycol, and water as well as their binary mixtures. *Industrial & Engineering Chemistry Research 51*, 21 (2012), 7428–7440. URL: https://doi.org/10.1021/ie300248z, arXiv:https://doi.org/10.1021/ie300248z, doi:10.1021/ie300248z. 7

[IAP18] Revised Release on the IAPWS Formulation 1995 for the Thermodynamic Properties of Ordinary Water Substance for General and Scientific Use, 2018. accessed 2019-09-23. URL: http://www.iapws.org/relguide/IAPWS-95.html. 5

[IMI*10] INGRAM S., MUNZNER T., IRVINE V., TORY M., BERGNER S., MÖLLER T.: Dimstiller: Workflows for dimensional analysis and reduction. In *2010 IEEE Symposium on Visual Analytics Science and Technology* (2010), pp. 3–10. doi:10.1109/VAST.2010.5652392. 2

[Koh98] KOHONEN T.: The self-organizing map. *Neurocomputing 21*, 1 (1998), 1 – 6. URL: http://www.sciencedirect.com/science/article/pii/S0925231298000307, doi:https://doi.org/10.1016/S0925-2312(98)00030-7. 2

[KTHRT*16] KRAUSE M., T. HENN A. M., R. TRUNK P. W., NATHEN P., KLEMENS F., MAIER M.-L.: Openlb release 1.0: Open source lattice boltzmann code. URL: http://www.openlb.net/download. 6

[LGZY16] LIU R., GUO H., ZHANG J., YUAN X.: Comparative visualization of vector field ensembles based on longest common subsequence. In *2016 IEEE Pacific Visualization Symposium (PacificVis)* (2016), pp. 96–103. doi:10.1109/PACIFICVIS.2016.7465256. 2

[Ma07] MA K. L.: Machine learning to boost the next generation of visualization technology. *IEEE Computer Graphics and Applications 27*, 5 (2007), 6–9. doi:10.1109/MCG.2007.129. 2

[OJ14] OBERMAIER H., JOY K. I.: Future challenges for ensemble visualization. *IEEE Computer Graphics and Applications 34*, 3 (2014), 8–11. doi:10.1109/MCG.2014.52. 2

[PBCR11] PRETORIUS A. J., BRAY M., CARPENTER A. E., RUDDLE R. A.: Visualization of parameter space for image analysis. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2402–2411. doi:10.1109/TVCG.2011.253. 2

[RKGC*17] RUTKAI G., KÖSTER A., GUEVARA-CARRION G., JANZEN T., SCHAPPALS M., GLASS C. W., BERNREUTHER M., WAFAI A., STEPHAN S., KOHNS M., REISER S., DEUBLEIN S., HORSCH M., HASSE H., VRABEC J.: ms2: A molecular simulation tool for thermodynamic properties, release 3.0. *Computer Physics Communications 221* (2017), 343 – 351. URL: http://www.sciencedirect.com/science/article/pii/S0010465517302527, doi:https://doi.org/10.1016/j.cpc.2017.07.025. 7

[SGL*16] SHU Q., GUO H., LIANG J., CHE L., LIU J., YUAN X.: Ensemblegraph: Interactive visual analysis of spatiotemporal behaviors in ensemble simulation data. In *2016 IEEE Pacific Visualization Symposium (PacificVis)* (2016), pp. 56–63. doi:10.1109/PACIFICVIS.2016.7465251. 2

[SHBP] SEDLMAIR M., HEINZL C., BRUCKNER S., PIRINGER H.: Visual parameter space analysis: A conceptual framework. *IEEE TVCG (Proc. InfoVis)*, 2014. 2

[SKB*18] SACHA D., KRAUS M., BERNARD J., BEHRISCH M., SCHRECK T., ASANO Y., KEIM D. A.: Somflow: Guided exploratory cluster analysis with self-organizing maps and analytic provenance. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (2018), 120–130. doi:10.1109/TVCG.2017.2744805. 2

[SMG*15] SPLECHTNA R., MATKOVIĆ K., GRAČANIN D., JELOVIĆ M., HAUSER H.: Interactive visual steering of hierarchical simulation ensembles. In *2015 IEEE Conference on Visual Analytics Science and Technology (VAST)* (2015), pp. 89–96. doi:10.1109/VAST.2015.7347635. 2

[US09] UNGER A., SCHUMANN H.: Visual support for the understanding of simulation processes. In *2009 IEEE Pacific Visualization Symposium* (2009), pp. 57–64. doi:10.1109/PACIFICVIS.2009.4906838. 2

[Ves99] VESANTO J.: Som-based data visualization methods. *Intell. Data Anal. 3*, 2 (1999), 111–126. URL: http://dx.doi.org/10.1016/S1088-467X(99)00013-X, doi:10.1016/S1088-467X(99)00013-X. 2

[WHLS18] WANG J., HAZARIKA S., LI C., SHEN H.: Visualization and visual analysis of ensemble data: A survey. *IEEE Transactions on Visualization and Computer Graphics* (2018), 1–1. doi:10.1109/TVCG.2018.2853721. 2

[WLSL17] WANG J., LIU X., SHEN H. W., LIN G.: Multi-resolution climate ensemble parameter analysis with nested parallel coordinates plots. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 81–90. doi:10.1109/TVCG.2016.2598830. 2

[WMK13] WHITAKER R. T., MIRZARGAR M., KIRBY R. M.: Contour boxplots: A method for characterizing uncertainty in feature sets from simulation ensembles. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2713–2722. doi:10.1109/TVCG.2013.143. 2

[WT] WANG C., TAO J.: Graphs in scientific visualization: A survey. *Computer Graphics Forum 36*, 1, 263–287. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12800, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12800, doi:10.1111/cgf.12800. 2