INTERACTIVE VISUAL PARAMETER-SPACE AND CORRELATION ANALYSIS OF SPATIO-TEMPORAL SIMULATION ENSEMBLES

MARINA EVERS

INFORMATIK

Dissertationsthema

Interactive Visual Parameter-Space and Correlation Analysis of Spatio-Temporal Simulation Ensembles

Inaugural-Dissertation
zur Erlangung des Doktorgrades (Dr. rer. nat.)
der Naturwissenschaften im Fachbereich
Mathematik und Informatik
der Mathematisch-Naturwissenschaftlichen Fakultät
der Westfälischen Wilhelms-Universität Münster

vorgelegt von

MARINA EVERS

aus Lingen (Ems)

- 2023 -

Dekan: Prof. Dr. Arthur Bartels
Erster Gutachter: Prof. Dr.-Ing. Lars Linsen
Zweiter Gutachter: Prof. Dr. Rüdiger Westermann
Tag der mündlichen Prüfung: ....
Tag der Promotion: ....

ABSTRACT

Numerical simulations are commonly used to model spatio-temporal phenomena across different domains, including physics, climate science, and medicine. The individual simulation results often vary over space and time but might also contain different fields. Simulations are run several times with different input parameters and initial conditions to capture their influence. The resulting set of simulation runs is called a simulation ensemble. Such simulation ensembles capture the uncertainty induced by the initial conditions or unknown parameter settings but can also be used to investigate how the parameters influence the simulation outcome. However, simulation ensembles contain large amounts of complex data, which makes them challenging to analyze. To obtain a comprehensive understanding of the data, visual analysis approaches combine automatic methods with the expert knowledge of the scientist.

This thesis presents interactive visual approaches targeting different facets of the analysis of simulation ensembles. Applications to simulation ensembles from the domains of physics, climate science, and medicine as well as feedback from experts in the respective domains show the utility of our approaches.

The first part of the thesis concentrates on exploring the parameter space to understand how different parameters influence the simulation output and identify the most important ones. An interactive method for semi-automatically partitioning the multi-dimensional parameter space into regions with similar simulation outcomes is proposed. This multi-dimensional partitioning can be investigated in a distortion-free visualization that allows for seeing the partitions' extent and obtaining a geometrical understanding. Furthermore, we also propose a 2D embedding that provides an overview of the partitioning and preserves its topology, the partitions' sizes, and boundaries. Interactive measure definitions support a flexible, domain-specific analysis of the parameter space. Coherent topological landscapes allow for understanding the topological variations in the dependency of the input parameters. While these approaches focus on qualitative parameter-space analysis, we introduce an interactive visual approach for analyzing the quantitative sensitivity in different spatial regions.

The second part of the thesis targets the analysis of correlations between different spatial regions to study, in particular, long-range phenomena in climate science like the El Niño phenomenon. Based on a summary visualization of the correlations covering the entire simulation ensemble, uncertainty-aware visual analysis approaches allow for investigating the correlations on different levels of detail.

ZUSAMMENFASSUNG

Numerische Simulationen werden genutzt, um räumlich-zeitliche Phänomene in wissenschaftlichen Bereichen wie Physik, Klimawissenschaften und Medizin zu modellieren. Die einzelnen Simulationsläufe variieren üblicherweise in Raum und Zeit, können aber auch mehrere Felder beinhalten. Um den Einfluss der Eingabeparameter und der Startbedingungen zu ermitteln, werden die Simulationen mehrere Male ausgeführt. Daraus ergibt sich eine Menge an Simulationsläufen, die auch als Simulationsensemble bezeichnet wird. Solche Simulationsensembles erfassen die Unsicherheit, die aus unbekannten Anfangsbedingungen oder Parametern stammt. Simulationsensembles beinhalten große Mengen komplexer Daten, die zu Herausforderungen in der Analyse führen. Um ein umfassendes Verständnis der Daten zu erhalten, kombinieren Ansätze der visuellen Analytik automatische Methodiken mit dem Expertenwissen der Wissenschaftler.

Diese Arbeit präsentiert interaktive visuelle Ansätze, die auf die Analyse der unterschiedlichen Facetten von Simulationsensembles abzielen. Die Anwendbarkeit der Techniken wird durch Beispiele aus der Physik, der Klimaforschung und der Medizin gezeigt sowie durch die Rückmeldung von Experten.

Der erste Teil der Arbeit konzentriert sich auf die Untersuchung des Parameterraums, um den Einfluss der Parameter auf das Simulationsergebnis zu verstehen und die wichtigsten Parameter zu identifizieren. Eine interaktive Methode zur halbautomatischen Partitionierung von mehrdimensionalen Parameterräumen wird vorgestellt, wobei die Partitionen basierend auf Ähnlichkeiten im Simulationsergebnis geformt werden. Die Partitionierungen können in einer verzerrungsfreien Visualisierung dargestellt werden, die ein Ablesen der Ausdehnung der Segmente und das Bilden eines geometrischen Verständnisses erlaubt. Außerdem schlagen wir eine 2D Einbettung vor, die einen Überblick über die Partitionierung vermittelt und sowohl die Topologie und die Größen der Partitionen als auch die ihrer Grenzflächen berücksichtigt. Die interaktive Definition abgeleiteter Größen erlaubt eine flexible, anwendungsspezifische Analyse des Parameterraums. Kohärente topologische Landschaften vermitteln ein Verständnis über die topologische Variation in Abhängigkeit der Eingabeparameter. Während sich diese Ansätze auf eine qualitative Parameterraumanalyse konzentrieren, addressiert ein in-

teraktiver visueller Ansatz die Analyse der quantitativen Parametersensitivität in unterschiedlichen räumlichen Regionen.

Im zweiten Teil der Arbeit werden Korrelationen zwischen unterschiedlichen räumlichen Regionen untersucht, was die Analyse langreichweitiger Phänomene aus der Klimaforschung, wie zum Beispiel des El Niño Phänomens, ermöglicht. Basierend auf einer Überblickvisualisierung, die die Korrelationen der kompletten Simulationsensembles darstellt, werden visuelle Analyseansätze unter Berücksichtigung der Unsicherheit vorgestellt. Diese erlauben es, die Korrelation auf unterschiedlichen Detailgraden zu untersuchen.

PUBLICATIONS

Parts of the content of this thesis are published or will be published in the following manuscripts:

- **M. Evers** and L. Linsen, 2D Embeddings of Multi-dimensional Partitionings, *to be submitted*
- **M. Evers**, S. Leistikow H. Rave, and L. Linsen, Interactive Visual Analysis of Spatial Sensitivities, *to be submitted*
- **M. Evers**, M. Böttinger and L. Linsen, Interactive Visual Analysis of Regional Time Series Correlation in Multi-field Climate Ensembles, *Workshop on Visualisation in Environmental Sciences (EnvirVis)*, 69-76 (2023)
- **M. Evers**, R. Wittkowski and L. Linsen, ASEVis: Visual Exploration of Active System Ensembles to Define Characteristic Measures, *2022 IEEE Visualization and Visual Analytics (VIS)*, 150-154 (2022)
- **M. Evers** and L. Linsen, Multi-dimensional Parameter-space Partitioning of Spatio-temporal Simulation Ensembles, *Computers & Graphics* 104: 140-151 (2022)
- **M. Evers**, M. Herick, V. Molchanov and L. Linsen, Coherent Topological Landscapes for Simulation Ensembles, In *Computer Vision, Imaging and Computer Graphics Theory and Applications*, edited by Bouatouch K. et al., 223-237 (2022)
- **M. Evers***, K. Huesmann* and L. Linsen, Uncertainty-aware Visualization of Regional Time Series Correlation in Spatio-temporal Ensembles, *Computer Graphics Forum* 40, No. 3: 519-530 (2021)
(* The authors contributed equally.)

The following work is not part of the thesis but was developed during the same period:

- **M. Evers**, A. Derstroff, S. Leistikow, T. Schneider, L. Mallepree, J. Stambke, M. Leisgang, S. Sprafke, M. Schuhl, N. Krefft, F. Droese and L. Linsen, Visual Analytics of Soccer Player Performance Using Objective Ratings, *under review*

- G. Borrelli, L. Hagemann, J. Steinkühler, A. Derstroff, **M. Evers**, K. Huesmann, S. Leistikow, H. Rave, R. Sabbagh Gol and L. Linsen, How Wildfires Spread and Why: Visual Multi-field Analysis of Vorticity-driven Lateral Spread Ensembles, *Proceedings of IEEE VIS 2022 (SciVis Contest)* (2022), Winning Entry

- K. Heimes, **M. Evers**, T. Gerrits, S. Gyawali, D. Sinden, T. Preusser and L. Linsen, Studying the effect of tissue properties on radiofrequency ablation by visual simulation ensemble analysis, *Eurographics Workshop on Visual Computing for Biology and Medicine* (2022), Honorable Mention

- **M. Evers** and R. Wittkowski, A colloidal time crystal and its tempomechanical properties, *arXiv preprint arXiv:2112.04498* (2021), *under review*

- **M. Evers**, S. Leistikow, A. Derstroff, T. Gerrits, K. Huesmann, J. Hollenbeck, J. Seljami and L. Linsen, Visual Analysis of Spatio-temporal Features in Multi-field Earth's Mantle Convection Simulations, *Proceedings of IEEE VIS 2021 (SciVis Contest)* (2021)

ACKNOWLEDGEMENTS

I want to thank everyone who supported me during the last four years, without whom this work would not have been possible.

First and foremost, I want to thank my supervisor Lars Linsen for his continuous support throughout the last years and from whom I learned a lot. I am very grateful that he provided me with opportunities to follow my research interests and had an open door to answer my questions.

I also want to thank all my colleagues and co-authors for many interesting and inspiring discussions. I want to especially thank Karim Huesmann for the productive joint work, Simon Leistikow for always answering technical questions, and Adrian Derstroff for the discussions in our office. Further, I want to thank Hennes Rave, Reyhaneh Sabbagh Gol, Maryam Saffo, Vladimir Molchanov, Gulsayyar Ali, Gabriel Borrelli, Tim Gerrits, Quynh Quang Ngo, José Matute Flores, and Muhammad Jawad. I really enjoyed my time in the group together with you. Special thanks go to Michael Böttinger for fruitful discussions about visualization in climate science and to Maria Herick for topological discussions. I would like to thank Verena Hörr, Andreas Völker, and Raphael Wittkowski for their valuable ideas and feedback from the domain experts' perspective. I am also very grateful for the great organizational support, first by Evelyn Egelkamp and then by Katharina Sichma.

Further, I want to thank my family and friends for supporting me outside of work. I especially want to thank Moritz Bensberg, who also supported me during long days in the home office.

I am truly thankful for everyone mentioned above and to all the others who made this thesis possible.

This work was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) grant 260446826 (LI 1530/21-2).

CONTENTS

I INTRODUCTION 1

1 INTRODUCTION 3
- 1.1 Simulation Ensembles 4
- 1.2 Ensemble Visualization 6
- 1.3 Contributions 9

2 ENSEMBLE VISUALIZATION: STATE OF THE ART AND BACKGROUND 13
- 2.1 Visualization and Visual Analytics 13
- 2.2 Ensemble Visualization 16
  - 2.2.1 Parameter-Space Analysis 18
  - 2.2.2 Uncertainty Visualization 20
  - 2.2.3 Multi-field Visualization 21
- 2.3 Common Techniques in Ensemble Visualization 22
  - 2.3.1 Dimensionality Reduction 22
  - 2.3.2 Multi-run Similarity Plot 26
  - 2.3.3 Parallel Coordinates Plot 29
  - 2.3.4 Scatterplot Matrix 30
  - 2.3.5 Functional Boxplots 30

II PARAMETER-SPACE ANALYSIS 33

3 SEGMENTATION EMBEDDING 35
- 3.1 Related Work 36
- 3.2 Fundamentals 37
  - 3.2.1 Orthogonal Graph Drawing 38
  - 3.2.2 Cellular Automata 40
- 3.3 Visualization Tasks 41
- 3.4 Overview 42
- 3.5 Segmentation Embedding 44
  - 3.5.1 Graph Embedding 44
  - 3.5.2 Cellular Automaton 45
  - 3.5.3 Segmentation Visualization 54
- 3.6 Algorithmic Evaluation 57
  - 3.6.1 Quality Criteria 57
  - 3.6.2 Results 58
- 3.7 Embedding of 3D Segmentations 68
- 3.8 Discussion 71

4 ANALYSIS OF PARTITIONINGS 73

4.1 Related Work 74
4.2 Task Definition 75
4.3 Overview 76
  4.3.1 Synthetic Dataset 79
4.4 Similarity Space Analysis 80
  4.4.1 Clustering 80
  4.4.2 Cluster Analysis 83
  4.4.3 Similarity Embedding 85
  4.4.4 Temporal Evolution Plot 86
4.5 Parameter-Space Visualizations 87
  4.5.1 Hyper-slicer 87
  4.5.2 Parameter-Space Embedding 94
4.6 Comparison to Alternative Visual Encodings 96
4.7 Analytical Workflow 98
4.8 Case Studies 100
  4.8.1 Blood Flow 100
  4.8.2 Semiconductor Quantum Wire 102
  4.8.3 Active Crystal 104
4.9 Discussion 106
5 INTERACTIVE DEFINITION OF CHARACTERISTIC MEASURES 109
5.1 Related Work 110
5.2 Requirement and Task Analysis 111
5.3 Workflow 113
5.4 Visual Analysis System 114
  5.4.1 Detail Visualizations 114
  5.4.2 Interactive Programming Interface 116
  5.4.3 Timeplot 117
  5.4.4 Heatmap 117
5.5 Analyzing Active Crystal Dynamics 118
5.6 Discussion 121
6 TOPOLOGICAL ANALYSIS OF PARAMETER DEPENDENCIES 123
6.1 Related Work and Background 124
  6.1.1 Topological Landscapes 126
  6.1.2 Merge Tree Matching 128
6.2 Overview 129
6.3 Coherent Contour Trees 130
  6.3.1 Distance Metric 130
  6.3.2 Matching 132
6.4 Coherent Visualization of Topological Landscapes 134
  6.4.1 Animation 135
  6.4.2 Static Visualization 136
6.5 Results 137

6.5.1 Synthetic Datasets 137
6.5.2 Pattern Formation in 2D 139
6.5.3 Cavity Flow in 3D 139
6.6 Discussion 141
7 SPATIAL GLOBAL SENSITIVITY ANALYSIS 145
7.1 Related Work and Background 146
7.1.1 Global Sensitivity Measures 147
7.1.2 Space-filling Curves 151
7.1.3 Horizon Graphs 153
7.2 Problem Specification 154
7.3 Overview 155
7.4 Visual Design 157
7.4.1 Parallel Coordinates Plot 157
7.4.2 Spatial Sensitivity Visualization 159
7.4.3 Parameter Dependency Visualization 164
7.5 Evaluation 166
7.5.1 Datasets 166
7.5.2 Comparison of Sensitivity Computation Methods . 168
7.5.3 Space-filling Curve 173
7.6 Usage Scenarios 174
7.6.1 Synthetic data 174
7.6.2 Radiofrequency ablation data 175
7.6.3 Aneurysm data 179
7.7 Discussion 181

III UNCERTAINTY-AWARE ANALYSIS OF CORRELATIONS 183
8 SIMILARITY IMAGES 185
8.1 Related Work 186
8.1.1 Watershed Segmentation 187
8.2 Problem Specification 188
8.3 Similarity Image 188
8.3.1 Time Series Ensemble Correlation 189
8.3.2 3D Embedding 190
8.3.3 Color Mapping 191
8.4 Hierarchical Segmentation 192
8.5 Results 194
8.5.1 Synthetic Dataset 194
8.5.2 Global Climate Simulation 197
8.5.3 3D Blood Flow Ensemble 199
8.6 Discussion 201
9 UNCERTAINTY-AWARE HIERARCHICAL CORRELATION ANALYSIS 203

9.1 Overview 204
9.2 Multi-level Correlations 204
9.3 Visual Design 207
    9.3.1 Region Visualization 208
    9.3.2 Correlation Heatmap 210
    9.3.3 Uncertainty-aware Time Series Visualization 212
    9.3.4 Coordinated Interactions 214
9.4 Results 215
    9.4.1 Synthetic Dataset 215
    9.4.2 2D Climate Ensemble 215
9.5 Discussion 218

10 INTERACTIVE CORRELATION ANALYSIS IN MULTI-FIELD CLIMATE ENSEMBLES 221
10.1 Workflow 222
10.2 Preprocessing 223
10.3 Visual Design 224
    10.3.1 UMAP Embedding 224
    10.3.2 Map View 226
    10.3.3 Heatmap 227
    10.3.4 Fourier Analysis 229
10.4 Synthetic Dataset 231
10.5 Climate Ensemble Analysis 233
    10.5.1 North-Atlantic Oscillation 233
    10.5.2 El Niño/Southern Oscillation 236
    10.5.3 Domain Expert Feedback 239
10.6 Discussion and Conclusion 240

IV CONCLUSION 243

11 CONCLUSION AND FUTURE WORK 245

V APPENDIX 251

A ADDITIONAL RESULTS 253
A.1 Comparison of LMDS and PMDS 253
A.2 Similarity Images 254
A.3 Evaluation of DGSA 258
A.4 Evaluation of Space-filling Curves 259
B DATASETS 261
B.1 Synthetic Dataset with 4D parameter space 261
B.2 Reaction-Diffusion System 262
B.3 Cavity Flow 262

BIBLIOGRAPHY 264

Part I

INTRODUCTION

1

INTRODUCTION

Scientific experiments in some areas, such as high-energy physics or astrophysics, can cost billions of euros. One prominent example is CERN which plans to build a new supercollider that will cost at least 21 billion euros [57]. Numerical experiments are often significantly cheaper and allow for investigating such phenomena beforehand to perform more targeted experiments that are more likely to show results of interest. Recently, the strong growth in computational resources available increased the number of cases where experiments can be replaced or preceded by numerical simulations, which led to significant growth in the popularity of numerical simulations [44]. Their computation allows for investigating different phenomena and developing an understanding to effectively reduce the number of experiments for solving a specific research question. Therefore, numerical simulations are used in different fields, including engineering, physics, biology, and medicine.

It is significantly easier to control external conditions in numerical simulations. For experiments, much effort is often required to exclude influences of the surrounding, such as changes in temperature, dust, or radiation. Sometimes they cannot be excluded at all. Not only can these factors be controlled explicitly in simulations, but it is also possible to vary the parameters freely. Radiofrequency ablations for medical treatment, for example, are highly dependent on the tissue properties, which are hard to measure and impossible to influence as they depend on the respective patient [151].

Sometimes simulations are the only possibility to investigate a particular phenomenon because controlled experiments are impossible, for example in climate science. The influence of the greenhouse gas emission in the future years cannot be investigated in experiments. However, simulation models allow for creating predictions about different possible scenarios. This allows for understanding processes that would otherwise be very hard to investigate using only experiments.

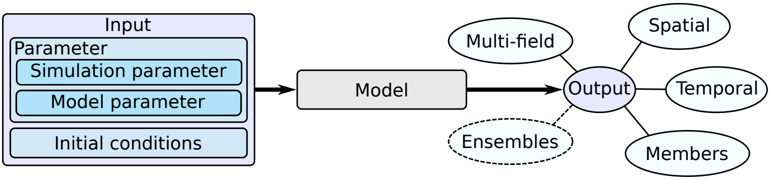


Figure 1.1: Numerical simulations can be described by considering the input, the model, and the output. Different kinds of parameters are used as input of a model, which produces an output including multiple different dimensions. This thesis does not cover the ensemble dimension visualized with a dashed outline.

However, a single numerical simulation does not allow to capture different parameter settings or initial conditions. Therefore, it is common to run several simulations by varying the input parameters, the initial conditions, or both. This set of simulations is then referred to as a simulation ensemble. Such ensembles may also cover the uncertainty in the output. Creating simulation ensembles leads to large amounts of data with many facets that are challenging to analyze. Especially in explorative analyses, a fully automatic evaluation procedure is commonly unavailable. Thus, visualizing the data allows for an analysis that combines the good visual perception of humans with the domain knowledge of the expert who created the simulation.

1.1 SIMULATION ENSEMBLES

We start by introducing the structure of the data obtained by numerical simulations and clarifying the terminology regarding the different facets of the data. The creation of an ensemble, as considered in this thesis, includes three different aspects, which are shown in Figure 1.1: the simulation *input*, the simulation *model*, and the simulation *output*.

The simulation input can be divided into two different groups of input. The *input parameters* are parameter values used to drive the simulation and can be divided into two groups. *Simulation parameters* influence the simulation and have a meaning in the domain. They can be either numerical or categorical values. Examples of this parameter type are tissue properties in medical simulations or the inlet velocities in flow field simulations. *Model parameters* form the other group of parameters. These parameters influence the calculations and include, for example, the step width used for numerical integrations or the grid size in simulation models covering two- or three-dimensional spatial domains. However, in this

Table 1.1: Dimensions of ensemble data covered in this thesis. As Chapter 3 discusses a general algorithm not specific to ensemble data, it is excluded from this table.

Chapter	Spatial	Temporal	Members	Multi-field
4	✗	✗	✗	
5		✗	✗	
6	✗	✗	✗	
7	✗		✗	
8	✗	✗	✗	✗
9	✗	✗	✗	
10	✗	✗	✗	✗

thesis, we mainly focus on investigating simulation parameters. If model parameters are considered, they are treated together with the simulation parameters such that we consider input parameters in general without further differentiation. Further, we focus on numerical parameters and do not cover categorical parameters and the corresponding challenges of heterogeneous data.

The second kind of input are *initial conditions*, which are used to initialize the simulation model. However, if the initial conditions are unclear, they can also be varied, leading to an ensemble structure. The variation of initial conditions allows for capturing the uncertainty and propagating it to the ensemble output. While it is common for input parameters to analyze the qualitative dependence of the output on the input, this is significantly less common for the analysis of initial conditions.

The simulation model contains the simulation code or program itself. In this work, we will treat the model as a black box that takes an input as described above and computes an output based on the input. While not all simulation models are deterministic, we do not cover the particular aspects of the analysis of non-deterministic models in this work.

The simulation model's output for several simulation runs based on different inputs is called the simulation output or a simulation ensemble. In general, the whole simulation output can contain up to five different dimensions as proposed by Wang et al. [360]. The first dimension is the *ensemble dimension* which is formed by multiple, different ensembles. For example, different models can lead to one ensemble for each model, which then need to be analyzed together. This work does not tackle this dimension but instead focuses on analyzing single ensembles. The second dimension is the *member dimension*. For each simulation input, one simulation result is created by a single run of the simulation code. We refer to this output as an ensemble member or run where these two terms are used exchangeably in this thesis. If the data contains several ensemble members, they form a separate dimension. This member dimension is the most important difference between traditional (not ensemble) scientific data corresponding to a single member and ensemble data, as we analyze it in this thesis. Third, ensembles may contain a *time dimension* which describes the variation over time. Especially dynamic phenomena can only be analyzed by considering their temporal evolution. The fourth dimension is the *spatial dimension*. Spatially resolved simulations are very common to model, for example, flow fields, climate data, or physical phenomena, with 2D or 3D being the most common spatial dimensionality. The spatial dimension is often covered in volume or grid data. However, this is only sometimes the case. The last dimension is the *variable dimension*. Each spatial location stores several variables for each time step and each simulation run. The variables can be scalars, vectors, or tensors. The data is called multi-variate, multi-faceted, or multi-field ensemble if different variables are stored. One example of a multi-field ensemble is climate data that stores temperature, pressure, wind, and precipitation fields.

Not all of the dimensions are present in each ensemble. For example, if the ensemble is static, which means it contains only a single timestep, the time dimension is considered to be omitted [360]. In this thesis, we focus on spatio-temporal multi-field simulation ensembles. We do not address the ensemble dimension and exclude specifics of vector or tensor data. A summary about the different dimensions covered in the different chapters is provided in Table 1.1.

1.2 ENSEMBLE VISUALIZATION

The interplay of the many different dimensions of ensemble data usually results in large amounts of data. Thus, the analysis of ensemble data is very challenging. Automatic and semi-automatic approaches can reduce the data, process it, and derive information that is easier to handle. However, purely automatic approaches are not always available, not always suitable for explorative analysis, and often not well suited for obtaining an intuitive understanding of the underlying data. At the same time, visualizations support developing an intuitive understanding. Humans are very good at finding patterns in data. Additionally, domain experts have much experience in their fields, which they bring into the data analysis and which allows them to reach decisions and find inter-

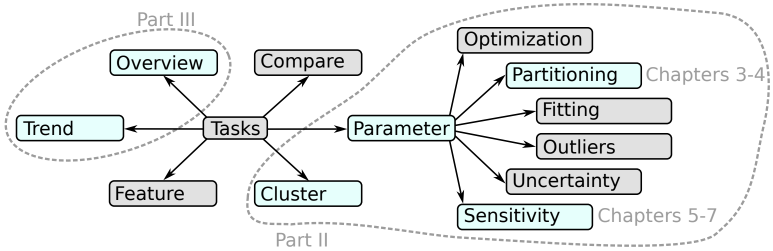


Figure 1.2: Overview of different tasks for the visual analysis of simulation ensembles. The tasks for ensemble analysis were identified by Wang et al. [360], while the tasks for parameter-space analysis were identified by Sedlmair et al. [308].

esting information. Therefore, visualizations can be used to support the domain scientists in their analysis.

Visual analytics combines the strengths of automatic approaches and visualizations to include the human in the analysis. Due to the complex nature of ensemble data, it is very difficult to effectively visualize it in one static visualization while also showing details. Interactive visualization approaches can solve the problem by allowing the users to interact with the visualization directly. In this way, they can, for example, explore the data in different levels of detail.

The complex task of analyzing a complete ensemble or specific aspects of this kind of data can be summarized by a set of frequent tasks as proposed by Wang et al. [360] (see Figure 1.2). One of the tasks is to obtain an *overview* about the ensemble. The goal is to create a visual summary representing the overall ensemble behavior, including all members. *Comparing* different ensemble members serves the purpose of identifying differences and similarities between simulation runs. This comparison task normally addresses pairwise similarities, but comparisons between several ensemble members are also possible. Another common task is *clustering* the simulation runs into groups that show a similar behavior. *Trend analysis* mainly targets the temporal aspect of the data. This goal targets the investigation of temporal trends of the whole ensemble, groups of ensemble members, or single ones. *Feature analysis* targets mainly spatial features even though temporal features are also possible. It is possible to analyze features in single ensemble members but also groups of members. The last task is the analysis of *input parameters* of the simulation ensemble. This task focuses on the input-output relations and is one of the core tasks tackled in this work.

A more detailed structure for analyzing the parameter space is presented by Sedlmair et al. [308]. Their conceptual framework for the parameter-space analysis contains six tasks, some related to the higher-level tasks identified by Wang et al. [360] for general ensemble analysis. The *optimization* task aims at finding a suitable input to obtain a desired outcome. If the desired outcome is clearly defined and quantified by a single number, automatic optimization strategies can be employed. However, if the desired outcome is unclear or contains competing objectives, human judgment aided by visualization becomes beneficial. A related task identified by Sedlmair et al. is *fitting* the simulation output to measured data. Typically, numerical simulations model real-world behavior, and one of its core targets is understanding the underlying real-world phenomena. Even though this task is similar to optimization problems, the user wants to approximate a particular result instead of finding the best result using a general optimization function.

Partitioning the parameter space divides it into regions with similar model behavior. This task is closely related to the clustering task, as identified by Wang et al. Usually, groups of ensemble members are identified based on the simulation outcome, which is why a clustering of the simulation result is used as a base to partition the parameter space. *Outlier* detection aims at finding simulation runs that are special in some kind and differ from most of the behaviors. This task typically includes a comparison between ensemble members (see general tasks by Wang et al.) and also holds for a more general ensemble analysis that does not explicitly target the input-output connections.

The member dimension of a simulation ensemble introduces *uncertainty* into the data that should be encoded visually. Uncertainty analysis includes the analysis of different sources of uncertainty, including the uncertainty in parameter choices as well as uncertain initial conditions. *Sensitivity* analysis is often seen as an aspect of uncertainty, but the goal differs. While uncertainty analysis generally asks about the certainty or reliability of the output, sensitivity analysis targets the influence of the different input parameters [239]. This includes quantitative sensitivity analysis (how large is the influence of the individual parameters?) as well as qualitative analysis (in which way do the parameters influence the outcome, for example, does an increase of the parameter value lead to an increase or decrease of the simulation outputs?).

In this work, we put a particular focus on the tasks of clustering ensemble members to analyze the parameter space (Part ii) and create an overview of the ensemble together with an analysis of the temporal trend (Part iii) where we also discuss uncertainty in the data. Regarding the parameter-space analysis, this work primarily focuses on the aspects of partitionings (see Chapters 3 and 4) and sensitivity analysis (see Chapter 5-7, where Chapters 5 and 6 target the qualitative sensitivity and Chapter 7 deals with quantitative sensitivity analysis). Our proposed approaches also allow for outlier detection and analysis as well as uncertainty investigation, but it is not the primary contribution.

1.3 CONTRIBUTIONS

This work addresses the interactive visual analysis of spatio-temporal multi-field simulation ensembles. The contribution of this thesis targets two topics, where the first focuses on different aspects of parameter-space analysis. The second one targets the analysis of spatial correlations in ensemble data which can be seen as a subtask of trend analysis as the temporal aspect plays a big role.

After presenting an overview of the literature and introducing common methods in ensemble visualization (see Chapter 2), this thesis contains the following contributions:

1. We propose embedding multi-dimensional segmentations or partitionings in the 2D plane (see Chapter 3). The embedding allows for obtaining an overview of a partitioning, which is, for example, helpful in analyzing parameter-space partitionings. The embedding is a method for providing an occlusion-free overview of the complete partitioning of multiple dimensions. It preserves the essential key features of the partitioning: the topology of the segmentation, segment sizes, and the length of the segment boundaries.

2. Interactive visual analysis approaches are proposed that focus on understanding the input parameters' quantitative and qualitative influences on the simulation outcome. Partitioning the parameter space allows finding regions of similar behavior in the simulation output (see Chapter 4). Additionally, understanding the dependency of the simulation output on the input parameters provides insights into the model and the underlying system. On the one hand, this process can be supported by interactively defining measures (see Chapter 5). On the other hand, analyzing the topological structure and its variations in the parameter space allows for identifying structural changes in the simulation ensemble (see Chapter 6). Sensitivity measures provide a quantitative metric for the parameter dependency and, thus, support the identification of the most relevant parameters which might vary over the spatial simu-

lation domain (see Chapter 7). In more detail, the contributions for parameter-space partitioning can be summarized as follows:

   a) For a distortion-free visualization of the parameter space and its partitionings while also showing the boundaries of the different segments, we propose an enriched hyper-slicer as presented in Chapter 4. Compared to the embedding presented in Chapter 3, the hyper-slicer allows to deduce the extent of the partitions in the individual dimensions of the parameter space and facilitates obtaining a geometrical understanding of the partitioning. The hyper-slicer is embedded in an interactive visual analysis tool with additional linked views that provide further context and facilitate navigation.

   b) We propose a workflow for interactively defining measures to understand how the simulation output depends on the input parameters in Chapter 5. The workflow is implemented in the interactive visual analysis tool ASEVis (Active System Ensemble Visualization) that we developed in the scope of a design study in the field of active systems.

   c) To study the dependency of the simulation output's topological structure on the input parameters, we present a method to create coherent contour trees in Chapter 6. Coherent contour trees can be visualized as coherent topological landscapes that allow for intuitive tracking of the topological variations and, for example, identifying the parameter values at which major changes in the output appear.

   d) Spatial sensitivities can be analyzed by a combined overview visualization using Horizon Graphs and space-filling curves to reduce occlusion as presented in Chapter 7. For this purpose, we generalize the concept of data-driven space-filling curves [399] to multi-field data. To investigate the qualitative sensitivity concerning the parameter space, we include a direct visualization of the simulation output on the input parameters. As the analysis result strongly depends on the chosen sensitivity and space-filling curve computation algorithms, we compare different approaches and derive a guideline on which approach to choose.

3. Our new analysis methods support the global visual analysis of correlations between different spatial regions:

   a) We propose an overview visualization called similarity images to analyze the spatial correlations in ensemble data. Similarity images use color coding to encode similarities in different spatial regions, see Chapter 8.

   b) Based on the similarity images, we present an uncertainty-aware interactive visual analysis approach with locally adaptable levels of detail in Chapter 9. The approach allows for studying time series correlations while including time lags in the analysis.

   c) We propose an interactive visual analysis approach based on dimensionality reduction to investigate correlations among spatial regions in multiple fields at once, see Chapter 10. The method allows for a comprehensive analysis including the correlation's spread of the ensemble and an uncertainty-aware investigation of the frequency spectrum.

4. All approaches are applied to simulation ensembles of different domains to demonstrate their utility.

Finally, we conclude with a discussion of the contributions and propose future research directions.

2

ENSEMBLE VISUALIZATION: STATE OF THE ART AND BACKGROUND

We start by presenting the current state of related work relevant to this thesis. Here, we provide a general overview. More specific topics related to single aspects are discussed directly in the corresponding chapters. First, we provide a high-level overview of the visual analysis process in Section 2.1. Section 2.2 provides more details on related work in ensemble visualization and its different facets. Section 2.3 presents background information on essential visualization techniques used on various points in this thesis.

2.1 VISUALIZATION AND VISUAL ANALYTICS

Visualization, which is located at the intersection between computer graphics and human-computer interaction, is the graphical representation of data that aims at supporting humans for the given analysis task [237]. Today, large amounts of data are created in various different fields, including medicine, physics, and geology, but also, for example, tracked by social networks [292, 175]. The ever-growing amount of data makes an analysis without using computers infeasible.

However, while computers allow fast data processing, humans are good at bringing their previous experience into the analysis process. Therefore, it is helpful to include domain experts in the analysis of data. Visually representing the data allows for obtaining an intuitive understanding while also helping to identifying patterns quickly. Visualizations are generally preferable to fully automated approaches if a process needs an in-depth understanding, cannot be automated, or the visualization approach acts as an intermediate step to develop new automatic methods. It takes advantage of the human's capability to process visual information in parallel and can reduce the cognitive load, which is beneficial to, for example, reduce the impact of the human's limited working memory [344]. In contrast to more comprehensive visualizations, statis-

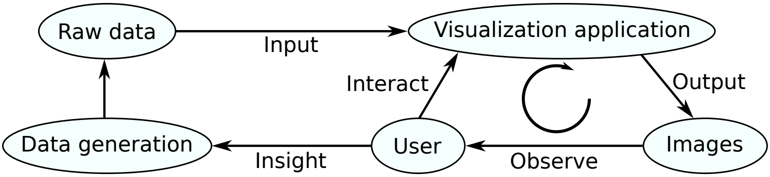


Figure 2.1: Interactive visualization process as described by Telea [336].

tical summaries of data often dramatically compress information which might lead to the loss of context or important data [16].

Visualization allows for showing larger amounts of data while still keeping them cognitively digestible for the user. Further, the user can still process and analyze the data visually even if the problem that should be solved is not well-defined [237]. This supports an exploratory analysis allowing to detect previously undefined patterns and, thus, formulating a new hypothesis of the data.

In the field of visualization, it used to be common to differentiate between information visualization (InfoVis) and scientific visualization (SciVis) [344, 188]. InfoVis commonly deals with data that does not contain spatial positions, such as tables, networks, or text data, and is discrete in nature. Scientific visualization, however, works on data with a spatial component stemming from different fields. This kind of data is often discrete but samples a continuous phenomenon. While these two kinds of data used to be treated separately, combinations of techniques from both fields become more common [188]. For example, the visualization of simulation ensembles contains spatial aspects for single time steps and members. However, to obtain overview visualizations and comparison of the members, techniques stemming from InfoVis research such as projection methods and parallel coordinates are used [360]. Even though all chapters in this thesis deal with scientific data, the proposed methods combine both fields.

Interaction plays a crucial role in handling the growing complexity and size of data. A static visualization commonly shows only a single level of detail or a certain single aspect of the data. Interaction, however, allows the user to change the view by interacting with it, allowing the use of different queries [237]. The general interactive visualization process as presented by Telea [336] is shown in Figure 2.1. It starts with data generation, which can have many different sources. For scientific data, these sources are normally either measurements or numerical simulations [87]. These data sources yield the raw data, which in the case of simulations, could cover the spatial and the temporal domain and con-

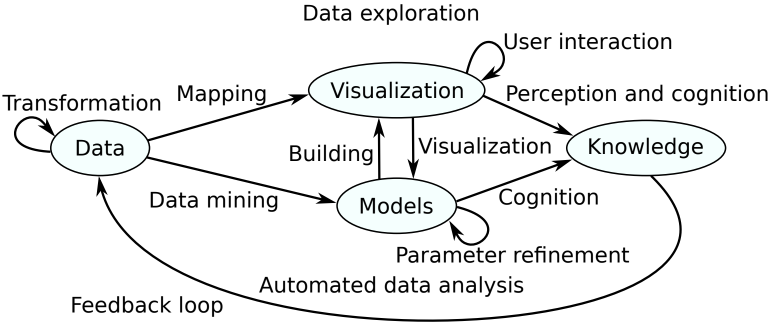


Figure 2.2: Visual analytics process as proposed by Keim et al. [185, 186]

tain multiple fields. This raw data is used as an input to a visualization pipeline. The output is an image, a set of images, or an interactive visualization. The end user then observes this set of (not necessarily static) images. In the context of this work, the end users are scientists. The users gain insights into the original phenomenon and interact with the visualization application to influence the output. Thus, visually analyzing data is usually an iterative process.

Visual analytics is a relatively young field of research defined by Cook and Thomas as "the science of analytical reasoning facilitated by interactive visual interfaces" [68, 338]. By combining computational methods with an interactive visual analysis executed by a human, the goal is to combine the advantages of both aspects of data analysis. It allows for analyzing more complex and often heterogeneous data. This combination of methods also leads to a strong interdisciplinarity of methods. Visual analytics solutions do not only combine methods from SciVis and InfoVis with methods from data science, statistics, and machine learning but also contain a strong connection to the data domain [186]. Developing an effective visual analytics approach without domain knowledge is nearly impossible, which is often solved by close collaborations with people working in the corresponding domain.

The visual analytics process described by Keim et al. [186, 185] is shown in Figure 2.2. The *data* which should be analyzed might first be transformed to bring it into a format that can be processed. It is then mapped to a *visualization* that can be investigated using the visualization pipeline as shown in Figure 2.1. Besides the data exploration, an automatic analysis is performed. Here, data mining methods are applied to a *model* which can be tuned by refining the model parameters. Note that the term model is used here broadly and includes, for example, dimensionality reduction techniques. In visual analytics, there is a strong connection between data exploration and automated analysis. Both kinds of analysis lead to a *knowledge* generation through cognition and perception. Normally, this process is rather a feedback loop than a linear process. Thus, the gained knowledge can again be used to work with the data and start the process again for further refinement or investigation of other aspects of the data.

One common navigation strategy in interactive visualizations is provided by the information seeking mantra proposed by Ben Shneiderman: "Overview first, zoom and filter, then details on demand" [313]. In visual analytics, this paradigm was extended by Keim at al. [186] to include the automatic analysis, which leads to "Analyze First - Show the Important - Zoom, Filter and Analyze Further - Details on Demand". Besides these rather general paradigms, several different taxonomies of interactions on different levels of detail in the context of visual analytics and information visualization have been proposed [392, 213, 65, 364]. One way of implementing these interaction and navigation strategies is by using multiple coordinated views [287].

The research process in visualization and visual analytics typically contains several different steps. One of the first steps is clearly understanding the problem. In the case of *design studies*, this often results in a task analysis and abstraction [309, 329]. A visual design can be developed based on the identified problem and tasks. This visual design can be formed by combining existing visualizations, which is common in design studies. In this case, the design space is explored, and design decisions are derived from the analysis tasks. If no suitable techniques and algorithms exist, new methods need to be developed [235]. In both cases, it is important to evaluate the visual design and the algorithm [236, 230]. If a user-centered design principle [344] is followed, this can include user studies, use cases, or collecting feedback from domain experts. In the case of algorithmic contributions, this includes, for example, run time analyses and evaluating performance metrics. The nested model presented by Munzner et al. [236] forms a general model for evaluating visualization research.

2.2 ENSEMBLE VISUALIZATION

With the growing amount of ensemble data available, various approaches for its analysis have been proposed. Approaches for the analysis of simulation ensembles include a wide range of application areas, including climate and weather data [243, 222, 200, 355, 396], ocean flow fields [386, 158, 279], medical data [151] and examples from chemistry [315], physics [203] and engineering [34, 226].

The state of the art in ensemble visualization and its various aspects have been covered in different surveys [219, 6, 281]. Obermaier and Joy [248] identified a set of mathematical, algorithmic, and conceptual challenges. While the mathematical challenges include defining features and metrics, the algorithmic challenges deal with data complexity and enable an exploration of the parameter space. The conceptual challenges include finding the best prediction, connecting the parameter space to the data, and intuitively visualizing it. These aspects are closely related to the identified challenges by Crossno [71]. Further open questions are presented by Wilson and Potter [374].

In this thesis, we focus especially on the connection between parameter space and similarity space and the exploration of the parameter dependencies and sensitivities. We also address the problem of feature definition by providing an approach for the interactive definition of characteristic measures that should preserve certain features. Broad state-of-the-art overviews about the current approaches of the field are published by Kehrer and Hauser [183], who generally address multifaceted scientific data, and Wang et al. [360], who provide a survey about the different facets of ensemble visualization.

A *similarity-based analysis* of simulation ensembles is a common approach that often includes clustering in different dimensions. Ferstl et al. [105, 106] presented approaches for clustering and visualization of ensemble data, focusing on weather data. Clustering can also be used to compare model output with reference data [195] or to show multi-resolution ensembles in nested parallel coordinates [361]. Climate ensembles can also be clustered hierarchically [181]. Further, clustering can be applied to visualize similar ensemble members in flow data [169]. Besides clustering, computing similarities between ensemble members allows for creating data projections on different levels of detail. Jänicke et al. [167] used a graph-based projection of the multi-dimensional space in which brushing can be used to define features. An overview of a simulation ensemble can be obtained by using a multi-run similarity plot as proposed by Fofonov et al. [110] where, for example, a field similarity measure [111] or an isocontour similarity measure [112] can be used to create a time-dependent similarity embedding. Section 2.3.2 provides more details about this approach.

Many approaches in the scientific visualization literature deal with the visualization of *time-dependent temporal data* [14, 6, 5]. Often, animations are used to tackle the temporal component [7], but other approaches try to incorporate the time-varying data in a single volume rendering by aggregating different time steps into a single volume [378]. However, the previously mentioned approaches do not deal with ensemble data. In the field of ensemble visualization, the use of graph structures may be used to show the temporal component of the data. Obermaier et al. presented a flow-graph representation of the ensemble based on clustering [247]. Shu et al. [314] also proposed a graph-based view, EnsembleGraph, to show the temporal evolution and embedded this approach in a visual analytics system. Another approach used shape comparisons, visualizations of cluster trees, and glyph-based visualizations to show spatio-temporal ensemble data [138].

Besides the temporal aspect, many different approaches have been used to visualize the different facets of ensemble data. Recently, *topological methods* for analyzing simulation ensembles became more common [21, 102, 152, 232]. A detailed discussion of the state of the art in topological ensemble visualization is provided in Chapter 6. Additionally, *deep learning* has been used to create surrogate models and for visualizing the data [147, 148, 327, 137, 311, 142]. For a comprehensive survey on the use of machine learning in the field of scientific visualization, we refer to Wang and Han [359].

Another frequently-used approach are *aggregations* of the different ensemble dimensions [300, 277, 182]. These aggregations reduce the complexity of the data but also lead to the loss of information. Glyph-based visualizations can also be used to show certain aspects of the ensemble and, for example, compare ensemble members [263] or combine them with scatterplot-based visualizations to show ensembles of 2D scalar fields [267]. *In situ visualizations* are sometimes combined with computational steering, allowing for influencing the simulation and, for example, adapting parameters while the simulation is running. Matkovic et al. [224] proposed a computational steering approach in the context of engineering simulations. Unger et al. [347] did not intervene in the simulations but focused on visualizing the simulation process. Other aspects common in the analysis of ensembles are the comparison with ground truth data created by measurements [38, 201] as well as model validation [266].

2.2.1 *Parameter-Space Analysis*

Parameter-space analysis is a key question in the exploration of simulation ensembles, often created by varying input parameters. The conceptual framework proposed by Sedlmair et al. [308] covers different facets of parameter-space visualization. A recent survey by Piccolotto et al. [265] provides an up-to-date overview of parameter-space analysis beyond spatio-temporal simulation ensembles.

Different approaches focus on the *local dependencies* of the simulation outcome on the parameter space, often focusing on optimizing the parameters. Bruckner and Möller [53] proposed a tool that facilitates the parameter choice for physical animations but does not include a direct visualization that relates the simulation output to the input parameters. Parameter-space visualizations are supported by the approach by Berger et al. [33], who included local visualizations to support the optimizations of parameter values by showing the neighborhoods of the selected points in the parameter space. InSituNet [148] uses deep learning for parameter-space exploration and also includes a local sensitivity analysis. Approaches specific to parameter spaces of other data types, like time series, cannot be directly applied to spatio-temporal simulation ensembles [93]. For allowing a domain-specific analysis of the relationship between input parameters and simulation output, we provide a method to interactively define characteristic measures and visualize them with respect to the input parameters in Chapter 5.

Other approaches cover a more *global analysis* of parameter spaces. Paraglide [34] is a system that includes several different methods for parameter-space analysis but focuses mainly on the sampling aspect. It has also been proposed to visualize the parameter space using a scatterplot matrix combined with additional scatterplots to investigate geoscientific data [346]. Luboschik et al. [221] combined the visualization of parameter space and simulation outcome by showing the parameter space with a color coding in a line but do not include a geometric visualization of the parameter space. Tuner [343] used hyperslices to visualize the parameter space for image segmentation. Fernandes et al. [104] designed glyphs to study region transitions between regions of approximately homogeneous behavior in the parameter space. A parameter-space analysis approach on multiple levels of detail was proposed by Splechtna et al. [321]. Other authors proposed global, projection-based approaches [320, 253] or target categorical parameter spaces [345]. However, none of the presented parameter-space analysis methods includes a full workflow for semi-automatic segmentation and geometry-preserving global visualization as we present in Chapter 4. Additionally, none of these methods provides a complete overview of a multi-dimensional partitioning which shows the partitioning's topological structure but also the sizes of the partitions and their boundaries like we present in Chapter 3.

A slightly different aspect is the analysis of sensitivities to the input parameters. While some authors proposed methods for local sensitivity analysis [148] or mainly focused on comparisons [8], others provided methods for creating surrogate models that facilitate the computation of sensitivity indices [311]. Neuhauser et al. [240] used piecharts for visualizing multiple sensitivity values on trajectory data. Biswas et al. [37] combined clustering of spatial, global sensitivity indices with map-visualization to analyze weather ensembles. However, their approach does not generalize to 3D data like our approach in Chapter 7.

2.2.2 *Uncertainty Visualization*

Simulation ensembles allow for propagating the uncertainty of the input to the simulation output. Johnson and Sanderson [176] discussed the importance of visualizing uncertainty in 3D visualizations. Many surveys of the last years covered different topics of uncertainty visualization and included different categorizations [254, 128, 276, 52, 40, 172]. Some overview articles focus on particular aspects of uncertainty visualizations like a taxonomy for uncertainty in medical data [286] or the application of uncertainty visualization in bioinformatics [369].

While uncertainty in 1D is commonly visualized using boxplots or variants thereof [40], there is no standard way of uncertainty visualization in higher dimensions. Kao et al. [180] proposed a visualization of 2D distribution datasets that can be seen as a way of generalizing boxplots to 2D. Summary plots [275] extended boxplots with additional statistical measures and are also generalized to 2D. Other authors study the variability of gradients in 2D [258] or use color codings for variations in the data [126]. Using glyphs for uncertainty is another option that has been studied [285, 210]. Dynamic volume lines [370] use a space-filling curve to unfold spatial data to one dimension. As the 1D representation is scaled based on variation in the data, it allows for quickly spotting regions of the highest variability in spatial data.

Some methods include encoding the uncertainty in traditional volume visualization approaches. Sakhaee et al. [294] proposed an approach for volume rendering for uncertain data. Many methods have also been proposed to visualize uncertain isosurfaces [145, 20, 262, 274, 19, 259, 273, 79, 78].

A relatively unexplored area is the identification of uncertain features. One method for extracting uncertain features uses copula functions in different spatial locations [144]. Recently, it has been studied how uncertainty can be included in dimensionality reductions which are also commonly used to show different aspects of simulation ensembles. Hägele et al. [135] proposed an uncertainty-aware multi-dimensional scaling that allows for projecting uncertainty in the high-dimensional space in lower dimensions and, thus, extended the uncertainty-aware principal component analysis presented by Görtler et al. [124].

However, the presented approaches do not target the uncertainty in correlations between ensembles of time series, as we discuss in Chapters 8-10 of the thesis.

2.2.3 *Multi-field Visualization*

The visualization of spatial multi-field data is a challenging task that has been addressed in various approaches. A good overview, also beyond the field of ensemble analysis, is given by Fuchs et al. [117] and He et al. [150]. Abstract visualizations [363] and system using multiple views to visualize different aspects [287, 130, 381] can be used in the analysis of multi-field data. As presented by Molchanov et al. [233], continuous projections provide an opportunity to reduce the dimensionality induced by the different fields. To link different visualizations, brushing techniques are commonly used [198, 82].

While volume rendering is a popular technique for single-field volume data, multi-field volume renderings allow for displaying multi-field data. However, finding a suitable transfer function is critical for quality even though it is a complex task. Other fusion-based techniques can facilitate the volume visualization of multi-field data [376]. Woodring et al. [379] used an operator for comparative multi-field visualizations that are also shown as a volume rendering. Another possibility is a visualization of multi-field datasets based on combining different rendering techniques [134, 141].

One option to study the relationship between different fields is the study of correlations among the different variables that can be either visualized directly [362, 60, 326] or in derived visualizations like the so-called multi-field graph as presented by Sauber et al. [301]. Gosink et al. [125] proposed a query-driven technique that works on the relationships between the variables. Deep learning has also been used to understand how the different fields related to each other [137].

Dimensionality reduction techniques allow for reducing the data complexity. Demir et al. [77] used a space-filling curve to linearize the 3D domain to support an occlusion-free visualization of multiple fields. This approach is closely related to the visualization of multiple sensitivity fields, as discussed in more detail in Chapter 7. Hazarika et al. [143] proposed using probabilistic principle component analysis as a base for a partitioning approach for multivariate data. Temporal multi-dimensional scaling (MDS) plots [165] can be used to visualize multivariate time series for detecting patterns. Fofonov et al. proposed a similarity measure between scalar fields that is also applicable to multi-field data [111] as well as MultiVisA, a framework for the interactive analysis of multi-field simulation data [110]. These approaches have also been applied to analyze the impact of astroids [203]. Clustering techniques are also applied to the analysis of multi-field data. Long et al. [217] visualized hierarchical clusters in multivariate data by combining tree visualizations with parallel coordinates, while He et al. [149] used bi-clusters to analyze multivariate scientific data. Clustering can further be used to investivate distributions in multi-field data [199].

Some single-field volume visualization and analysis techniques have been extended to work with multi-field data. Feature level sets [168] are a generalization of isosurfaces, while Carr et al. [55] presented contour nets which generalize contour trees. Multi-field data can also be encoded in more abstract visualizations. Kumpf et al. [198] proposed enhanced violin plots to represent different ensemble members including multiple fields. Multi-field graphs [301] use a graph visualization for single multi-field volumes while Tao et al. [334] presented matrices of isosurface similarity maps for time-varying multivariate data. Different glyph designs have also been proposed in the context of multi-field visualizations [206, 289, 190].

Many of these approaches focus on single aspects of the multi-field data, and most of them do neither tackle the unique challenges imposed by ensemble data due to the multi-run nature nor the temporal aspect of the data. Therefore, especially those two aspects need further research. In Chapter 10, we contribute to addressing these challenges by investigating correlations between multiple fields of spatio-temporal simulation ensembles.

2.3 COMMON TECHNIQUES IN ENSEMBLE VISUALIZATION

In the following, we will explain different techniques commonly used in the analysis of ensemble data and relevant to different chapters of the thesis. The techniques only used within a specific chapter are introduced directly in that particular chapter.

2.3.1 *Dimensionality Reduction*

Dimensionality reduction is a group of techniques applied to reduce the dimensionality of multi- or high-dimensional data. In the scope of this work, the goal is typically a visualization of the data. Therefore, the target dimensionality is commonly 2D or 3D, as these dimensionalities are the most convenient to visualize. However, dimensionality reduction typically leads to a loss of information. Therefore, also higher target dimensions can be used in the context of visualization, see, for example, the visualizations using scatterplot matrices in Chapter 5.

Many different techniques for dimensionality reduction have been proposed. They optimize different criteria and are also differentiated by their properties. One commonly distinguishes between methods that preserve the local or the global structure and between linear and non-linear techniques.

In this section, we will provide a brief introduction to principal component analysis (PCA), multi-dimensional scaling (MDS), and uniform manifold approximation and projection (UMAP), which are the dimensionality reduction methods that are used in this thesis. For a more extensive discussion and evaluation of dimensionality reduction techniques, we refer to the comprehensive survey papers [98, 385, 291, 214, 187].

2.3.1.1 *Principal Component Analysis*

PCA aims at finding the principal directions of the dataset, which cover the largest variability among the data points. The principal directions or principal components can then be used to change the basis of the representation. PCA can reduce the dimensionality by only using the first components covering the largest variability. The principal components are the eigenvectors of the data points' covariance matrix [349]. This can be expressed as solving the eigenvalue problem

$$\mathrm{Cov}(\mathbf{X})\mathbf{M} = \boldsymbol{\lambda}\mathbf{M}\,,$$

where $\mathbf{X}$ is a $N \times d$ matrix of N standardized, d-dimensional data points. $\mathrm{Cov}(\mathbf{X})$ is a $d \times d$ covariance matrix, $\boldsymbol{\lambda}$ denotes the vector of eigenvalues, and $\mathbf{M}$ is a matrix of eigenvectors that form the principal components. Depending on the analysis task, the data points should be standardized before computing PCA. Standardization can be achieved by subtracting the mean and dividing by the standard deviation. The eigenvalues λ_i with $i = 1, \ldots, d$ can be used to estimate the percentage of the variance covered in the corresponding principal direction given by the i-th eigenvector. If only the k largest eigenvectors are considered, they can be used to form a $d \times k$ matrix $\mathbf{M}_r$. Then, low-dimensional points $\mathbf{Y}$ can then be computed as $\mathbf{Y} = \mathbf{X}\mathbf{M}_r$.

2.3.1.2 *Multi-dimensional Scaling*

MDS [373] is a dimensionality reduction technique that minimizes stress defined as

$$\phi(\mathbf{Y}) = \sum_{i,j} \left(d_{i,j} - \|\mathbf{y}_i - \mathbf{y}_j\|\right)^2 ,$$

where $d_{i,j}$ is the dissimilarity between the high-dimensional points $\mathbf{x}_i$ and $\mathbf{x}_j$ and $\mathbf{y}_i$ and $\mathbf{y}_j$ are their projections. In this work, we only consider classical MDS, which assumes that the dissimilarities $d_{i,j}$ contain metric properties. Non-metric MDS can be used for similarity measures that do not hold this assumption. PCA and classical MDS are equivalent if the Euclidean distance is used for the dissimilarity $d_{i,j}$. However, the advantage of classical MDS over PCA is its general applicability to other metric measures like field similarity measures (see Section 2.3.2).

Classical MDS can also be computed using an eigenvector decomposition [373]. However, the first step is creating a matrix of squared proximities:

$$\mathbf{P}^{(2)} = [d_{i,j}^2] .$$

Then, a method called double-centering is applied to obtain a matrix $\mathbf{B}$, which is used for the eigenvalue decomposition:

$$\mathbf{B} = -\frac{1}{2}\mathbf{J}\mathbf{P}^{(2)}\mathbf{J} ,$$

where $\mathbf{J} = \mathbf{I} - \frac{1}{N}\mathbf{E}$ using the identity matrix $\mathbf{I}$, the number of samples N and the matrix $\mathbf{E}$ with all entries being 1. Next, the m largest eigenvalues $\lambda_1, \ldots, \lambda_m$ of matrix $\mathbf{B}$ are extracted as well as m corresponding eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_m$. This can be achieved either by computing the full eigenvalue decomposition or, if this is infeasible due to the high dimensionality of the data, an iterative procedure can be used. The positions $\mathbf{Y}$ in the m-dimensional space can then be computed by

$$\mathbf{Y} = \mathbf{V}_m \mathbf{\Lambda}_m^{\frac{1}{2}}$$

where $\mathbf{V}_m$ is the matrix composed by m eigenvectors and $\mathbf{\Lambda}_m$ is the diagonal matrix with m eigenvalues of $\mathbf{B}$ on the diagonal.

Due to the singular value decomposition, MDS can be costly to compute for large datasets. In these cases, the MDS result can be approximated by embedding a subset of the sample points and approximating the embedding of the remaining points based on the already embedded points. Landmark MDS (LMDS) and Pivot MDS (PMDS) are two methods that follow this strategy.

LMDS [316] follows the idea of placing a subset of the sample points as landmarks. Thus, the eigendecomposition only needs to be computed on the landmarks, while the other points are placed as a linear combination of the landmark points. While classical MDS is based on the eigenvalue decomposition of the $n \times n$ matrix $\mathbf{B}$, LMDS reduces the size of the matrix to $k \times k$ for k samples. However, LMDS does not use additional non-landmark points that might already have been placed. PMDS [48] aims at overcoming this potential drawback by including the distances to the other points. PMDS achieves this goal by considering an $n \times k$ submatrix of the matrix $\mathbf{P}^{(2)}$. After double-centering the submatrix, one obtains a matrix $\mathbf{C}$. To obtain the positions of the sample points, the largest eigenvectors of $\mathbf{C}^T\mathbf{C}$ are computed. These two steps induce additional computational costs compared to LMDS, while the other steps require the same run time. Indeed, we find PMDS to yield better results, as shown by a brief comparison in Appendix A.1 and, thus, use it as an approximation for MDS in Section 8.5.3.

2.3.1.3 *UMAP*

While PCA and classical MDS are linear, distance-preserving dimensionality reduction techniques, nonlinear techniques allow for finding other features in the data. Many different nonlinear dimensionality reduction techniques exist, including locally linear embedding (LLE) [290], ISOMAP [337] or Kernel PCA [302].

T-distributed stochastic neighbor embedding (t-SNE) [348] is a nonlinear dimensionality technique optimized for showing clusters. However, as t-SNE only considers the samples' neighbors, the dataset's global structure is not preserved in the embedding.

Uniform manifold approximation and projection (UMAP) [228] allows for reducing the dimensionality based on finding Riemannian manifolds in the multi-dimensional space and, then, embedding it in the lower dimensional space. While providing comparable results in representing clusters in the embedding as t-SNE, UMAP tends to preserve better the global structure of the dataset, and can be computed more efficiently. In the following, we will provide a short description of the basic ideas of the algorithm. For a more detailed explanation, we refer to McInnes and Healy [228].

The first step of computing UMAP is approximating the manifold by creating a fuzzy graph whose vertices are formed by the points. For creating the edges, each point's k nearest neighbors are identified based on a user-defined value k. Here, small values of k result in depicting more local structures in the final projection, while large values of k lead to a more global representation. Next, a region around the point is created. This region is defined by a radius set based on a distance to the kth nearest neighbor. If these regions of two different points overlap, the points are connected by an edge of the graph, where the edge is weighted based on the distance between the points. This can be interpreted intuitively by a decrease in the probability of the edge if the distance between points increases. To ensure that each point is connected to at least its nearest neighbor, the edge weight is defined such that the probability only decreases beyond the nearest neighbor. Thus, the graph creation is governed by the number of considered nearest neighbors k.

After approximating the manifold by a graph, this graph needs to be embedded in the target domain. For this purpose, a force-based graph layout is used. In principle, a random initialization can be used, but starting from the spectral layout has shown to provide faster convergence. Then, the weighted edges are used to relax the layout and compute an embedding of the graph that approximates the high-dimensional graph as well as possible. UMAP is well suited to identify clusters which is why we use it in Chapter 10 to select groups of correlated segments.

2.3.2 *Multi-run Similarity Plot*

The multi-run similarity plot is an overview visualization of a whole ensemble originally proposed by Fofonov et al. [112]. Each spatio-temporal ensemble member is represented by a polyline where each point represents a single timestep connected in the temporal order. Proximity in the multi-run similarity plot encodes the similarity between the respective timesteps.

To compute this similarity-preserving visualization, at first, similarities between the individual time steps need to be computed. The isocontour similarity [112] compares the shapes of isosurfaces among two scalar fields for a given isovalue. As the isocontours can be seen as two sets defined by the spatial samples inside the isocontour, two different isocontours can be compared by the Jaccard distance, a standard distance measure for comparing sets. Since every ensemble member typically contains a large number of spatial samples, a Monte Carlo sampling of the spatial domain is used to reduce the amount of data. The sample points obtained by the Monte Carlo sampling are stored in a feature vector for the respective scalar field. For each sample point, it is decided if it lies inside the isocontour defined by the selected isovalue or outside. This results in a binary feature vector for each scalar field that can then be used to compute the Jaccard distance between two feature vectors A and B representing two scalar fields:

$$d(A, B) = 1 - \frac{M_{A \wedge B}}{M_{A \vee B}},$$

where $M_{A \wedge B}$ is the number of samples inside both isocontours and $M_{A \vee B}$ is the number of samples located within at least one of the isocontours.

Another similarity measure that can be used for creating multi-run similarity plots is the multi-field similarity measure [111], which is a generalization of the isocontour similarity that takes the whole field into account and does not require the choice of an isovalue. The expressibility of the isocontour similarity measure strongly depends on the choice of the isovalue, and even if a suitable isovalue was found, a single isocontour does not necessarily describe the underlying field sufficiently well. When considering all possible isocontours, one could notice that for each entry of the feature vector defined by the Monte Carlo sampling, there is precisely one isovalue for which it switches from inside the isocontour to outside the isocontour. This value corresponds to the scalar value stored in this spatial sample point.

For the computation of the multi-field similarity measure, we assume that the field values in the scalar fields are normalized to the range $[0, 1]$. Therefore, also the values a_i and b_i, which are the i-th entry of the vectors A and B, are normalized to this range. Then, we can compute the distance between vectors A and B as

$$d(A, B) = 1 - \frac{\sum_{i=1}^{N} (1 - \max(a_i, b_i))}{\sum_{i=1}^{N} (1 - \min(a_i, b_i))},$$

where N is the number of sample points chosen for the Monte Carlo approach. This similarity measure can be easily generalized to multi-field data, for example, by concatenating the feature vectors for the different fields. However, in this work, it is only used for processing individual scalar fields.

For creating a multi-run similarity plot, one of these field similarity measures is chosen to create a distance matrix between all timesteps of all ensemble members. This distance matrix can then be used as an input to the classical MDS algorithm as described in Section 2.3.1. In the projection, all samples of a single ensemble member are connected in temporal order. This leads to one polyline representing one ensemble member. An example of a multi-run similarity plot is shown in Figure 2.3.

The projection could be performed with different target dimensions. While a 3D visualization can show the most information of the original

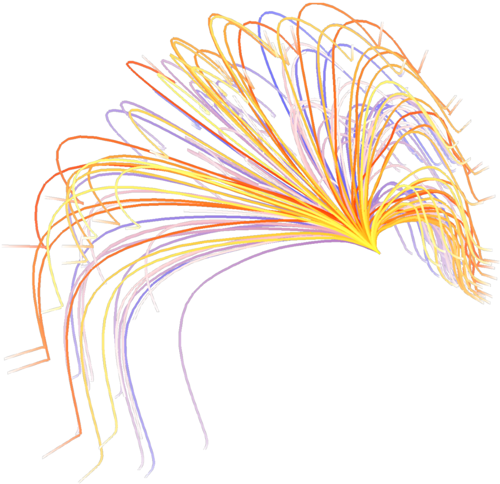


Figure 2.3: 2D multi-run similarity plot. Each polyline represents the temporal evolution of a single ensemble member where the colors differentiate between the members, and the saturation encodes time. Distances in this plot depict the similarities between the underlying scalar fields. All simulation runs originate at the same point but diverge over time.

dataset, 3D visualizations might be hard to understand, and the depth perception in the third dimension might be difficult. These problems do not occur in a 2D projection. A projection to the most important dimension allows several additional visualizations, as the second dimension can be used to show an additional variable. One example is the visualization of the evolution over time, where the horizontal axis of the visualization is used to encode time, and the vertical axis corresponds to the most important component of the data. Interaction with the visualization allows, for example, for selecting ensemble members or certain timesteps. If the 1-dimensional projection is visualized over time, it is also possible to select the time range of interest for further analysis.

In this thesis, we use multi-run similarity plots to obtain an overview about the spatio-temporal evolution of the ensemble members. We also extent the methods to aggregate over time such that each point represents an entire ensemble member which allows to identify groups of ensemble members as discussed in Chapter 4.

2.3.3 *Parallel Coordinates Plot*

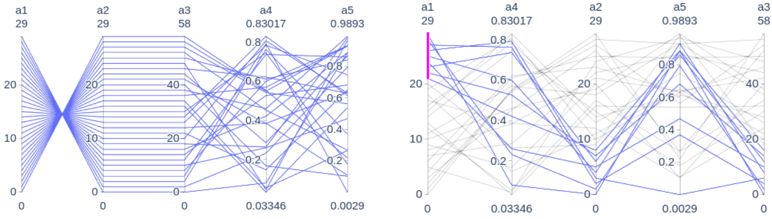
Parallel coordinates plots (PCP) allow for a distortion-free visualization of multi-dimensional data [153, 163, 173]. Each dimension is represented by an axis. Therefore, for a d-dimensional space, d axes are needed. Typically, they are placed in parallel. Each d-dimensional point is then visualized as a polyline where the vertices lie on the axes and represent the corresponding values.

If straight lines are used to connect the vertices on the axes, parallel coordinates exhibit a point-line-duality. Each point in the 2D Euclidean (sub)space is represented as a line segment in parallel coordinates. At the same time, each point in parallel coordinates is represented by a line in the 2D Euclidean space. Additionally, the lines between two adjacent axes allow for deriving information about the correlation of the attributes. Examples of positive, negative, and no correlation are shown in Figure 2.4.

However, this information can only be derived for adjacent axes. There is no inherent order of the axes, but it plays an important role in this visualization. To cover this problem, a common interaction supported by parallel coordinates is the possibility of changing the axes ordering interactively. In this way, the user can try different combinations and, thus, perform different pairwise analyses without losing the context of all dimensions.

In addition to identifying correlations among different dimensions, PCP can be used for finding and visualizing clusters [174]. Several applications and extensions have been proposed that, for example, allow for visualizing large amounts of data [245] by using focus+context techniques. Edge bundling is another option for dealing with high numbers of sampling points [205] while GPU-based approaches address scalability issues [324]. However, some limitations of PCP will be discussed in Chapter 4.

In this work, PCPs are used to visualize spatial multi-field data. In this case, spatial samples are considered as d-dimensional data points where d corresponds to the number of fields available. While PCPs allow for visualizing multiple values of the different scalar fields, they do not provide spatial context. This can be overcome by allowing for brushing on the axes and linking the selection to a spatial rendering. Besides simple brushing, a wide range of brushing techniques has been proposed to facilitate the selection of sample points for different contexts [198, 288, 140].



(a) (b)

Figure 2.4: Parallel coordinates for a 5-dimensional dataset. a) Between the first two axes, we see an anti-correlation. The second and third axes are correlated, while the other axes show no correlation. b) After reordering the axes, the patterns are hidden. The highlighted polylines in this visualization are selected by brushing on the first axis.

2.3.4 *Scatterplot Matrix*

A scatterplot matrix (SPLOM) provides an alternative to visualize multi-dimensional data. Here, n-dimensional data is shown as an $n \times n$ matrix, with each matrix cell containing the scatterplot for a pair-wise combination of dimensions as shown in Figure 2.5. Thus, the SPLOM contains all pair-wise combinations of the different dimensions and, therefore, also allows for finding pair-wise correlations. However, in contrast to the PCP, all pair-wise combinations are shown simultaneously, requiring no interaction to study different combinations. The diagonal of the matrix can be used to show the parameter names instead of plotting one dimension against itself. As the matrix is symmetric, it is also possible to only show its part above (or below) the diagonal in order to avoid redundancy.

One downside of the SPLOM is its limited scalability. As the screen space required scales quadratically with the number of dimensions that should be shown, the SPLOM is limited in the number of dimensions that could be displayed. One option to reduce the dimensionality and also identify multi-dimensional patterns in the data, is applying PCA and only showing the most relevant principal directions as discussed in more detail in Chapter 5.

2.3.5 *Functional Boxplots*

The temporal variation at a spatial sample point of a simulation run can be described as the discrete sampling of a continuous function. Functional boxplots [328] generalize classical boxplots to functional data. To achieve this goal, they use analogous statistical measures. Thus, one ob-

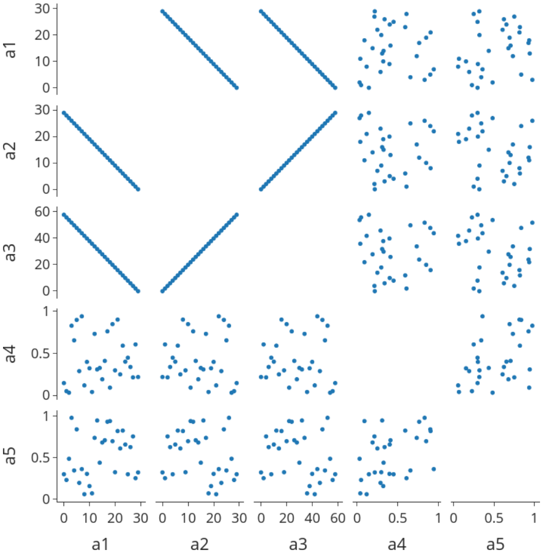


Figure 2.5: The scatterplot matrix for the same 5-dimensional dataset as shown in Figure 2.4 consists of scatterplots for all pair-wise combinations of dimensions.

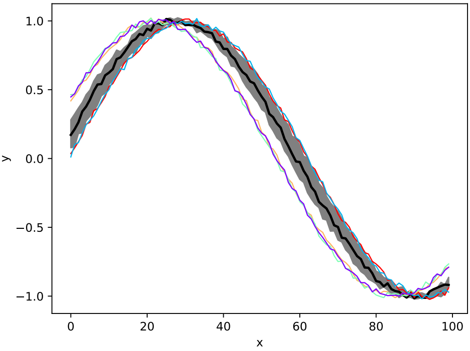


Figure 2.6: The functional boxplot provides an analogy to classical boxplots for functional data, where band depth is used for ordering the functions. The black line represents the median, the gray band is the 50% interval, and the colored lines are outliers.

tains a representation that considers the whole functional data. Compared to alternatives, such as computing point-wise boxplots, functional boxplots show features present in the data by considering functions over the entire domain.

The first step is ordering the functions to compute the different statistical values. This can be done using the concept of band depth or a modified band depth [218]. After computing the band depth of each function, they are reordered by decreasing band depth. The first function, which has the highest band depth, can be considered the most central curve. For functional boxplots, this curve is used as the median curve. The first half of the functions ordered by decreasing band depth is then used to define the 50% band of the data and, thus, is analogous to the box in classical boxplots [328]. The 50% region can be visually shown as an envelope around the data. In traditional boxplots, the whiskers usually represent the 1.5 interquartile range corresponding to the 50% region. Therefore, extending this concept to functional boxplots is straightforward by extending the envelope of the 50% region with a factor of 1.5. Then, each function outside this region can be considered an outlier.

Figure 2.6 shows an example of a functional boxplot where ten sine curves with random noise are used. This figure already shows the sensitivity to outliers as all functions outside of the 50% band are outliers. While this problem can be reduced by increasing the factor for determining outliers, choosing an optimal factor for a given dataset is a nontrivial task. In Chapters 9 and 10, we build upon functional boxplots to show the uncertainty in the temporal evolution of different climate variables.

Part II

PARAMETER-SPACE ANALYSIS

3

SEGMENTATION EMBEDDING

Segmentations are commonly used in a wide variety of fields ranging from image segmentations in medical imaging over segmentations of climate data (see Chapter 9) to multi-dimensional parameter-space partitionings (see Chapter 4). While segmentations in two dimensions can be visualized easily, for example, by color coding the different segments, analyzing partitionings in three or more dimensions is challenging. Already in 3D, occlusion prevents obtaining a global overview of the complete segmentation. For visually analyzing segmentations of any dimension, the sizes and neighborhood relations of the different segments are of particular interest, for example, to study transitions between different regions of parameter spaces. Especially in data with more than three dimensions, understanding the structure of the segmentation is very challenging.

Our proposed method supports the visual analysis of parameter-space partitionings. Parameter spaces are commonly multi-dimensional. When partitioning the parameter space in regions of similar behavior in the simulation outcome, it is of interest how these regions are related to each other and which region covers which part of the parameter space. This also allows for identifying potentially interesting transitions between the parameter-space segments. For example, in flow simulations one commonly differentiates between laminar and turbulent flow. These regions and their transitions could be investigated in more detail in the following steps.

We propose an algorithm for embedding multi-dimensional segmentations in two dimensions while preserving the neighborhoods and the segment and boundary sizes. For the embedding, we create a graph structure of the segmentation, which we then embed in the 2D plane. The resulting graph drawing is an initial configuration for an optimization algorithm that adapts the segment and boundary sizes while preserving the graph's topology. As a result, we obtain an embedding preserving the multi-dimensional features while avoiding occlusion. We evaluate the influence of the different parameters in the optimization function based on a set of quality criteria for the resulting embedding. Additionally, we propose a rendering of the result that facilitates the analysis and highlights the most important features of the embedding. As proof of concept, we apply our approach to visualize 3D spatial domain segmentations. For applications to other real-world datasets, we include the segmentation embedding into a visual analytics tool and directly embed it in the analysis workflow, see Chapter 4. In Chapter 9, we also apply it for analyzing correlated regions in 3D ensemble data. Note that depending on the domain, either the term segmentation or partitioning is more common. In this chapter, these terms are used interchangeably.

After discussing related work to our approach in Section 3.1, we provide background on graph drawing and cellular automata in Section 3.2. After a task definition in Section 3.3, an overview of our algorithm is presented in Section 3.4 and details in Section 3.5 followed by an algorithmic evaluation, see Section 3.6, the application to 3D segmentations, see Section 3.7), and a discussion of the approach, see Section 3.8.

The results presented in this chapter are based on the following manuscript:

> **M. Evers** and L. Linsen, 2D Embeddings of Multi-dimensional Partitionings, *to be submitted*

Both authors contributed to discussing ideas, writing, and editing the manuscript. I developed and implemented the algorithm and created the results.

3.1 RELATED WORK

Segmentations of 3D volume data commonly occur in various domains, including medical imaging. The segmentation is commonly visualized using 2D slices or showing only a subset of the segments [278]. However, showing only individual segments or small sets of segments does not provide a global overview of the segmentation structure. While volume renderings allow for visualizing some segmentations using suitable transfer functions and opacity to reduce the occlusion problem, they do not scale well to a larger number of segments. For higher-dimensional spaces that, for example, occur as parameter spaces for simulation ensembles, the visualization becomes even more difficult. Different approaches for visually analyzing parameter spaces are discussed in Section 2.2.1. However, none of the approaches provide a global overview of the segmentation structure.

Segmentations can be represented as graphs, as proposed by Ren et al. [284]. The authors aim at visualizing joint layouts for segmented meshes but do not include the properties of the segments, like sizes. To visualize the graph derived from the segmentation, one of various graph drawing techniques [332, 357, 227, 354, 108, 81, 304, 155] can be selected. Among the most common methods are force-directed layouts [179, 115], layered graph drawing [325] and orthogonal layouts [27]. These techniques all visualize graphs as node-link diagrams.

As alternative graph visualizations, space-filling techniques have been proposed. Weighted trees can be encoded as treemaps [312]. Treemaps cannot be applied to our problem as the graphs derived from the segmentation are not trees. Techniques focussing on visualizing clusterings like Bubble-Sets [67], LineSets [11], and KelpFusion [229] are also related to our problem. These methods show regions around vertices while GMap [119, 160] creates a map-like layout that might be fragmented. To overcome this problem, either the clustering or the layout can be modified [193]. However, the topology is not necessarily preserved such that there are not always common boundaries if there are connecting edges between two clusters. A similar problem is targeted by MapSets [92], which has the disadvantage of causing complex regional layouts. The clustered graph visualization method by Wu et al. [382] deals with vertices belonging to more than one cluster by drawing connections on top of the layout. As these connections differ from the visualization of the clusters, they also induce perceived differences. For our approach, edge crossings in non-planar graphs should not be specially highlighted, which is why this visual encoding is not desirable for our problem.

Like our approach, cartograms aim to preserve areas for a given topology [246]. Usually, the layouts in the context of cartograms are planar and do not contain crossings, as they commonly occur in our cases if the original dimension is at least 3D. There are also methods for planar graphs that encode neighborhoods as shared boundaries [9, 391] but do not generalize to non-planar graphs. Overall, none of the discussed approaches is suitable for visualizing a multi-dimensional embedding, which can be described as a (potentially non-planar) graph where areas and boundary sizes are preserved in the visual representation.

3.2 FUNDAMENTALS

Our approach strongly builds on orthogonal graph drawing and cellular automata. Therefore, in this section, we briefly introduce the fundamentals of our work.

3.2.1 *Orthogonal Graph Drawing*

An undirected *graph* $G = (V, E)$ is defined by a finite set of vertices V and edges E where each edge $e = \{u, v\} \in E$ consists of an unordered pair of vertices $u, v \in V$. A *graph drawing* is the embedding of a graph in a 2D plane. A *planar graph* is a connected graph that can be drawn without edge crossings. A graph drawing of a planar graph divides the drawing plane into different regions, which are referred to as *faces*. Each face can be uniquely described by the sequence of adjacent edges. The external face of the graph is the only unbound face as it is only bound by edges to the inside. Thus, it corresponds to the outside of a 2D graph drawing.

Orthogonal graph drawings are characterized by allowing only vertical and horizontal edges [86]. These kinds of drawings have the advantage of having a relatively clean layout due to the few edge directions and angles of 90°. Orthogonal graphs are, for example, applied to very large-scale integration (VLSI) design to optimize the layout of the wires. Standard optimization criteria include minimizing the number of edge bends or edge crossings where the latter is relevant to our approach.

The so-called topology-shape-metrics approach minimizes the number of edge crossings and also keeps the number of edge bends low [331, 27, 333, 80]. As edge crossing minimization is an $\mathcal{NP}$-hard problem, this approach provides a suitable approximation for the graph drawing. This group of algorithms uses a three-step approach:

1. *Planarization:* The first step defines the topology of the graph drawing where two drawings have the same topology if one drawing can be obtained from the other by continuous deformation without altering the edge order.

2. *Orthogonalization:* This step defines the shape of the drawing. Two graph drawings have the same shape if one can be obtained from the other by modifying only the lengths of the edge segments.

3. *Compaction:* The last step defines the metrics, namely the coordinates of the vertices and edge bends. Two drawings contain the same metrics if they are equivalent up to rotation and translation.

The planarization step mainly aims at minimizing the number of edge crossings. As it is executed first, the minimization of edge crossings is the priority of the graph drawing algorithm. In the following, we will discuss the different steps and the relevant algorithms in more detail.

3.2.1.1 *Planarization*

The result of the planarization step is a planar embedding [80] which is an embedding of a graph without any edge crossings. This can be achieved for non-planar graphs by replacing the edge crossings with dummy vertices. Note that a planar embedding is a combinatorial description where the sequence of edges is defined for each face, but the vertices do not yet contain coordinates.

To obtain such a planarization layout and minimize the number of crossings, one can start by computing the maximum planar subgraph and then reinserting the remaining edges. In this step, each crossing is replaced by a dummy vertex.

Computing the maximum planar subgraph is also an $\mathcal{NP}$-hard problem. A range of approximation algorithms exist, but as we tested two of them, these will be discussed in more detail.

Triangle-based algorithm: This algorithm is based on computing the maximum number of triangles, which are three connected vertices, in a given graph and building a planar subgraph based on them [58]. In the first step of the algorithm, as many diamond subgraphs as possible are identified where a diamond consists of two triangles sharing an edge. In the second step, as many triangles as possible are added to the subgraph. This procedure might lead to unconnected subgraphs, which are finally connected to each other. It can be shown that this algorithm leads to an approximation factor of $\frac{13}{33}$ for the maximum planar subgraph problem.

PQ-tree algorithm: Alternatively, the maximum planar subgraph can be computed with the help of a PQ-tree [170] where a PQ-tree is a data structure that represents permutations. Even though the theoretical worst-case running time of this algorithm is $\mathcal{O}(|V|^2)$, it is usually very fast in practice [332]. This also allows for running the algorithm multiple times to improve the outcome of the algorithm. In each of the runs, different orders for inserting the edges to the tree can be used.

Different methods exist for reinserting the additional edges that were not part of the planar subgraph. They can be categorized into two groups: insert edges directly or reinsert stars which are vertices including incident edges. A recent study showed that mixed insertion works best [63]. The mixed insertion attempts to reinsert at least one endpoint of each edge that still needs to be included in the planar subgraph via star reinsertion. For this purpose, the corresponding end vertex is removed and reinserted including all incident edges. However, only vertices that are not cut vertices are chosen for reinsertion to avoid unconnected graphs. A cut vertex is defined as a vertex whose removal would lead to more connected components in the graph. If both vertices of the edge to be inserted correspond to cut vertices, the edge is inserted using the traditional edge insertion method.

Planarized graphs can be embedded using a Boyer-Myrvold planar embedding [47]. The basic idea of the algorithm is to add the edges one by one while preserving the planarity of the graph. As a result, one obtains a combinatorial embedding, an embedding where the order of edges is defined for each vertex. However, the vertices of a combinatorial embedding do not have positions. While the different faces in the embedding are defined by the edge order, it does not contain a designated external face which needs to be defined for a unique definition of the embedding.

3.2.1.2 *Orthogonalization and Compaction*

To define the shape of the graph drawing, a list of angles is computed for each edge, but the vertices still do not contain coordinates. One way of computing the orthogonalization implemented in the Open Graph Drawing Framework (ogdf) [62] is based on the bend layout minimizing algorithm proposed by Tamassia [331]. However, as the original algorithm only works on graphs with a maximum vertex degree of 4, the algorithm implemented in ogdf is adapted to work on general planar (or planarized) graphs. This algorithm uses the flow through the network to find a representation with a minimum number of bends. The high-degree nodes are placed inside boxes to generalize the algorithm to vertices with a degree higher than 4, and the edges are routed toward the incident position on the box.

Besides computing the coordinates for vertices and edge bends, the dummy vertices introduced in the planarization step are removed and replaced by edge crossings in the final compaction step. One option for the compaction step is computing the minimum-cost flow in the dual graph [191].

3.2.2 *Cellular Automata*

Cellular automata can be used for different applications, ranging from modeling in statistical mechanics [375] over biological modeling [97] to the computation of cartograms [84]. In this chapter, a cellular automaton is used to optimize the segments and boundary sizes. A cellular automaton consists of three different parts [64]:

1. A regular lattice of cells.
2. A set of states Φ. Each cell has exactly one state at each timestep.

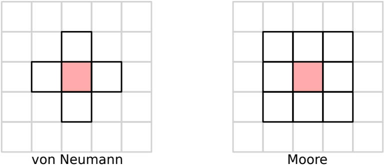


Figure 3.1: The von Neumann neighborhood and the Moore neighborhood of the central cell (red) are marked with a black outline.

3. A set of rules $\mathcal{R}$ which define the transition between states based on a given cell's neighborhood in the previous timestep. In the case of probabilistic cellular automata, a rule can also contain a random term.

While different neighborhood definitions can be considered, the most common ones in 2D automata are the von Neumann and the Moore neighborhoods, as shown schematically in Figure 3.1. The von Neumann neighborhood only consists of the four closest neighbors and is commonly known as the 4-neighborhood in computer graphics. The Moore neighborhood also includes the four neighbors on the diagonals and corresponds to the 8-neighborhood in the field of computer graphics. We refer to Chopard et al. [64] for a more detailed discussion of cellular automata.

3.3 VISUALIZATION TASKS

The visual encoding presented in this chapter should represent a multi-dimensional segmentation with its most important features. As we target the analysis of simulation ensembles, we derive tasks based on analyzing multi-dimensional parameter spaces as an application example. We identify the following set of tasks that the final visualization should fulfill:

T3.1 *Preserving the topology*. The topology of the segmentation is defined by neighborhood relationships between the segments. To understand the segmentation of the multi-dimensional space, it is important to preserve the topological structure in the low-dimensional embedding. For example, in parameter-space analysis, the topological structure directly reveals between which segments transitions are possible.

T3.2 *Representing the segment sizes*. The sizes of multi-dimensional segments might vary significantly. Therefore, they should also be represented in the lower-dimensional embedding. When analyzing a multi-dimensional parameter space, the segment sizes encode which portion of the parameter space corresponds to the respective segment. Assuming that the segmentation of the parameter space is created based on the similarity of the simulation output, the segment sizes also convey how frequently the characteristic behavior for the corresponding segment occurs. Visualizing the segment sizes allows, for example, to see whether all segments are similar in size or whether there are one or several segments that are significantly larger or smaller than the rest.

T3.3 *Representing the segment boundary sizes*. The boundaries between segments allow for transitions between the different segments. For example, in the case of parameter spaces, the sizes of the boundaries indicate how large the parameter ranges are that allow for direct transitions between the two considered segments. Additionally, the sizes of the joint segment boundary indicate how related the corresponding segments are and, thus, enrich the topological information. Based on the encoded segment sizes, it is possible to estimate whether there is a strong connection between the segments, which corresponds to a large boundary size, or the segments are only slightly connected. This information is also useful when interpreting the topology of the segmentation.

Note that we do not consider the shapes of the segments as representing multi-dimensional shapes in a single 2D embedding is infeasible. In the following, we will discuss how we address these visualization tasks.

3.4 OVERVIEW

The algorithm presented in this chapter provides a 2D visualization of a multi-dimensional segmentation or partitioning. We preserve the topology (task T3.1) by using a graph. Its embedding is optimized using a cellular automaton to encode the segment and boundary sizes (tasks T3.2 and T3.3). The different steps of the algorithm are shown in Figure 3.2.

We consider a segmented, n-dimensional volume where each connected component is considered as its segment labeled with a unique ID. Assuming a discrete sampling of the multi-dimensional domain, each spatial sample point is assigned to exactly one segment. We assume that the volume is sampled on a regular grid. Otherwise, a regular grid can be obtained by resampling.

The structure of the segmentation S can be described as a weighted, undirected graph $G = (V, E)$ with vertices V and edges E. Each vertex

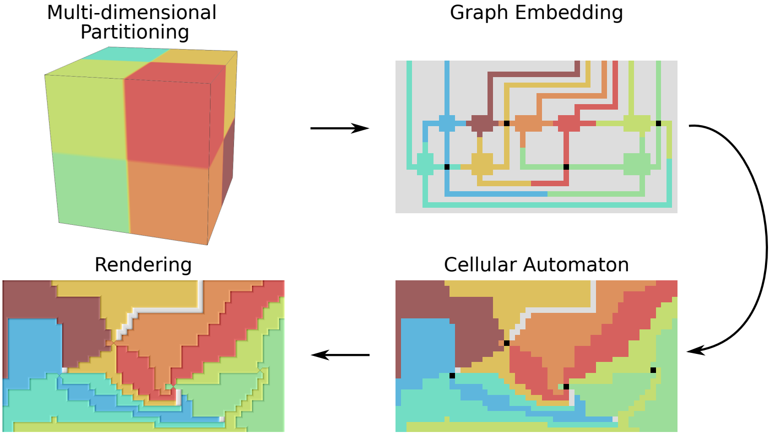


Figure 3.2: A multi-dimensional partitioning can be represented as a graph. Starting from the embedding of this graph, a cellular automaton approach is used for preserving the areas and boundary sizes. Finally, we render the different segments to visually encode the different features of the embedding, including segment boundaries, separating regions, and edge crossings.

$v_i \in V$ represents a segment $s_i \in S$. An edge $\{v_i, v_j\} \in E$ corresponds to a shared boundary between segments $s_i, s_j \in S$. The border of the original domain also influences the topological structure. To consider it, we insert an additional vertex v_b representing the outside of the domain. An edge $\{v_i, v_b\} \in E$ is added to the graph for each segment s_i connected to the outer boundary to preserve the topology.

Using the regular grid structure of the input data, we define two segments to share a boundary if and only if at least two grid cells of the corresponding segments are von Neumann neighbors. Thus, for a 3D domain, these cells share a face and cells that only share a vertex or an edge are not considered to have a shared boundary. One main goal of our approach is preserving the areas and boundary sizes. To include this information in the graph, we include the size $A_{s_i,nD}$ of a segment s_i as a weight for vertex v_i. The segment's size $A_{s_i,nD}$ is computed by counting the number of grid cells that belong to segment s_i. Each edge $\{v_i, v_j\}$ is weighted by the size of the boundary $L_{(s_i,s_j),nD}$, which corresponds to the number of cells that are neighbors in the sense of the von Neumann neighborhood.

The graph G is embedded in 2D, where we aim at minimizing edge crossings (see Section 3.5.1). This graph drawing preserves the segmentation's topology but not the area and boundary sizes. Therefore, we apply a cellular automaton for which we use the graph drawing as an input (see Section 3.5.2). The cellular automaton preserves the topological structure encoded by the graph but optimizes the representation for also encoding the area and boundary sizes to fulfill the visualization tasks T3.1 to T3.3. To visually encode the resulting embedding, we apply a rendering described in Section 3.5.3.

3.5 SEGMENTATION EMBEDDING

Based on the graph G representing the segmentation, we can compute a 2D embedding of the segmentation that fulfills tasks T3.1 to T3.3.

3.5.1 *Graph Embedding*

A graph embedding is used as the initial configuration for the cellular automaton. To keep the visualization as simple as possible, we want to minimize the edge crossings in the embedding. In addition to making the graph representation and the final visualization more complex, edge crossings represent an embedding artifact that is not present in the multi-dimensional space. As the problem of minimizing the number of edge crossings is $\mathcal{NP}$-complete [121], we use the planarization approach described in Section 3.2.1 as an approximation to obtain an orthogonal graph drawing.

Even though force-based graph drawings often show relatively low numbers of edge crossings, they do not explicitly aim at minimizing them. Thus, even drawings of planar graphs might contain edge crossings [122]. While additional vertices like the boundary vertex can be included by adding constraints, this might further complicate the layout.

An orthogonal layout containing only horizontal and vertical edges facilitates the rasterization required to obtain the initial configuration for the cellular automaton. For example, diagonal edge crossings would otherwise induce many special cases to be considered. After embedding the graph, we remove the boundary vertex to maintain only the vertices representing segments. To ensure that the initial configuration for the cellular automaton contains the same topology as the multi-dimensional segmentation, we connect all vertices with an edge to the boundary vertex to the outer boundary of the domain.

We use the Open Graph Drawing Framework (ogdf) [62] to compute the graph embedding. This library provides an implementation of the topology-shape-metrics approach where each step can be easily customized. We mainly follow the approach discussed in Section 3.2.1, using the algorithms presented in that section. The choice of the planar subgraph significantly influences the quality of the graph drawing. We compare the triangle-based approach to the approach using PQ-trees in Section 3.6. However, our graph drawing should follow the constraint that the boundary vertex should lie on the external face to facilitate the connections between the neighboring vertices and the boundary. Otherwise, the number of edge crossings might increase significantly. The external face can be defined after a combinatorial embedding is computed. To realize the constraint, we define the external face as one of the faces that contains the boundary node. The number of faces containing the boundary vertex equals the degree of this vertex. Therefore, we could obtain different embeddings that fulfill the criterion above.

Due to space constraints, connecting the vertices representing the segments to the domain's boundary might introduce additional edge crossings. Thus, we compute all embeddings where the boundary lies on the external face and choose the one minimizing the number of crossings.

3.5.2 *Cellular Automaton*

Based on the previously defined graph embedding, we want to create a visual representation that also preserves the areas of the segments and the sizes of the common boundaries. More concretely, we want to optimize the following criteria:

C3.1 Expand the graph vertices $v_i \in V$ to 2D segments whose relative area corresponds to the vertex weight $A_{s_i,nD}$ that represents the size in the multi-dimensional space (see task T3.2).

C3.2 Optimize the size of the boundary representing edge $e_{ij} = \{v_i, v_j\} \in E$ such that the length of the joint boundary between the 2D segments representing v_i and v_j corresponds to edge weight $L_{(s_i,s_j),nD}$ and thus to the boundary size in multi-dimensional space (see task T3.3).

As optimizing areas is closely related to cartograms, we build our approach on the cellular automata cartogram approach presented by Dorling [84]. Building upon a cellular automaton-based approach gives us a large flexibility for adapting it to our optimization criteria, especially preserving the boundary sizes. As we do not have any initial shapes to preserve as in traditional cartograms, it is not an issue that Dorling's approach does not preserve shapes [246]. Preserving shapes of n-dimensional segments embedded in the 2D space is infeasible.

3.5.2.1 *Initial Configuration*

We need to define the three parts of a cellular automaton, as described in Section 3.2.2. We first need to determine the grid resolution for defining a regular lattice of cells. For preserving the topology of the graph, it is important that the edges are separated by at least one cell. Here, we use the extent of the graph embedding, assuming a size of 20×20 cells for the vertices and a minimum separation of 20 cells between the vertices. A scaling factor f provides enough space for all edges and allows for a separating cell between the different edges. As we only use one border vertex v_b, it might have a high degree. Therefore, we use the degree $\deg(v_b)$ of this vertex to heuristically set the scaling factor to $f = \max(2, \sqrt{\deg(v_b)})$. The square root prevents the factor from growing too quickly because the number of cells grows quadratically with this factor, and a large number of cells increases the computation times and memory consumption significantly. In addition to the lattice used to create the visualization, we add a border of cells around it, representing the external boundary of the original domain.

Second, we need to define a set of states Φ. For this automaton, we define the possible states as the segmentation indices, which we assume to be strictly positive. Additionally, the state 0 represents cells not assigned to any segment and, in the following, referred to as background cells. The state -1 is assigned to the other cells representing the boundary and therefore defines the boundary conditions of the cellular automaton. Our approach can be directly applied to arbitrary domain shapes of the input data by labeling all grid cells belonging to no segments with -1. Cells representing edge crossings get assigned the state -2. For simplicity, we will use s_i for the segment and its ID. Thus, we obtain $\Phi = \{s_i | i = 1, ..., |S|\} \cup \{0, -1, -2\}$ where $|S|$ denotes the number of segments.

We can use a naive line drawing algorithm to assign cells to the edges of the graph drawing, as the orthogonal graph drawing, by definition, does not contain diagonal lines. Each edge gets the width of one cell. For routing the edges, we use the bending points, which are also provided by the graph drawing of the ogdf-framework. We start by drawing the edges connecting the vertices representing segments and, thus, do not involve the border vertex. Out of all line segments that are drawn for a given edge $\{v_i, v_j\}$, the cells belonging to the first half of the segments get assigned the ID of the segment corresponding to vertex v_i as a state while the second half gets assigned the ID corresponding to vertex v_j. The central line segment is split between both states. Here we apply a simple heuristic instead of more sophisticated methods like splitting the edge by arc length because we provide an initial state for a following optimization which is not sensible to the exact division of the edges. If an edge crossing is noticed when drawing the edges, the corresponding crossing cell gets assigned the state -2. Treating edge crossings separately allows us to preserve the topology in the relaxation phase. After drawing these edges, we initialize the cells belonging to each vertex except the boundary vertex by assigning the corresponding segment ID to a square of cells.

Finally, the connections to the domain boundary, which are represented by the edges $\{v_i, v_b\}$, need to be included. All cells belonging to this edge get assigned the ID corresponding to v_i. By default, we draw an edge by first following the edge routing until the boundary of the region, where the vertex representing v_b would be drawn, is reached. From here, we connect to the domain boundary of the cellular automaton's cell lattice to ensure that each segment that is connected to the domain boundary in multi-dimensional space is also connected to the domain boundary in the 2D embedding. This commonly works well because v_b was placed on the exterior face by definition of the graph embedding. An offset is added to the last bending point to avoid overlaps in the edge drawings. This offset varies between the edges leading to the border. However, if vertex v_i is closer to the cellular automaton's boundary than two times a vertex's size, no other vertex could be placed between v_i and the boundary. This is the case because we set the minimum distance between the vertices to the same value as the spatial extent of the vertices. Thus, we can directly connect v_i to the nearest boundary.

Vertices or edges going in the same direction as the current drawn edge provide obstacles that an edge crossing cannot overcome. For these cases, the edge is re-routed by introducing an offset of 2 cells until the edge's destination can be reached. However, this procedure might not always lead to a possible solution. One option would be opting for a significantly more complex edge routing. As this should be avoided, we instead neglect this graph drawing and choose one of the remaining drawings created by considering all faces including the boundary vertex v_b as an external face. Among all embeddings that correctly preserve the topology, we choose the one minimizing the number of edge crossings.

To reduce the number of cells, we remove all duplicate rows and columns of cells in a post-processing step. An example for the initial configuration of the cellular automaton after removing duplicates is presented in Figure 3.3. The last part of the cellular automaton definition is the definition of a set of rules. The derivation of these rules will be detailed in the following.

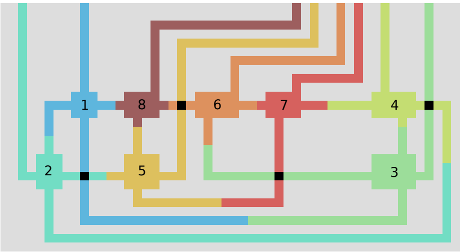


Figure 3.3: The graph drawing provides the initial configuration for the cellular automaton. Cells not belonging to any segment get assigned the ID 0 (light gray), while cells representing edge crossings get assigned the ID −2 (black). The cells belonging to segments get assigned the corresponding segment ID (shown as numbers) as a state.

3.5.2.2 *Optimization Criteria*

The optimization criteria, namely the preservation of areas (C3.1) and boundary sizes (C3.2), govern the definition of the set of rules. The deviation $d_{A,s}$ describes the deviation of the segment's relative size in the 2D embedding from its relative size in the multi-dimensional space. Thus, the deviation $d_{A,s}$ is minimized to optimize for area preservation (C3.1). For segment s, the area deviation is defined as

$$d_{A,s} = \frac{A_{s,nD}}{A_{nD}} - \frac{A_{s,2D}}{A_{2D}}, \tag{3.1}$$

where $A_{s,nD}$ denotes the size of segment s in the n-dimensional space, A_{nD} corresponds to the overall number of cells in the n-dimensional domain, $A_{s,2D}$ is the number of cells assigned to segment s in the cellular automaton and A_{2D} the total number of cells included in the cellular automaton and, thus, the total 2D domain size. If $d_{A,s} > 0$, the segment s needs to expand, while $d_{A,s} < 0$ indicates that the corresponding segment needs to shrink to represent the relative size of the segment accurately.

The deviation is computed for each segment individually. Thus, we can define the first rule as follows:

R3.1 A cell in state s_i should change its state if at least one cell in its von Neumann neighborhood is in state s_j and $d_{A,s_j} > d_{A,s_i}$.

As a second criterion, we aim to optimize the deviations between the relative boundary sizes (C3.2). In the following, we refer to the boundary sizes as boundary lengths to avoid confusion with the segment sizes. Nevertheless, these sizes are only lengths in 2D but, for example, areas in 3D and volumes in 4D. Note that the boundary length deviation is not unique to a segment but depends on the pair of segments that form the boundary. Thus, the boundary length deviation between segments s_i and s_j can be defined as

$$d_{L,(s_i,s_j)} = \frac{L_{(s_i,s_j),nD}}{L_{nD}} - \frac{L_{(s_i,s_j),2D}}{L_{2D}} , \tag{3.2}$$

where $L_{(s_i,s_j),nD}$ is the boundary length between segments s_i and s_j in the nD space, L_{nD} is the sum of all boundary lengths in nD, $L_{(s_i,s_j),2D}$ denotes the boundary lengths between s_i and s_j in the 2D embedding and L_{2D} is the sum of all boundary lengths in the 2D space. Analogous to the sizes, $d_{L,(s_i,s_j)} > 0$ means that the boundary length needs to increase while it should decrease for $d_{L,(s_i,s_j)} < 0$.

For defining a rule on when a cell should change, it needs to be considered that we obtain one deviation value per pair of segments that share a boundary. Each cell has four von Neumann neighbors, so there could be up to four boundary deviation values per cell. Additionally, changing the state of the cell does not necessarily change the boundary between the corresponding segments. To consider these two points, we compute for each neighboring segment how much the boundary length $L_{(s_i,s_j),2D}$ would vary if the state of the cell changes from s_i to s_j. The change in boundary length can be computed as

$$\Delta L_{(s_i,s_j)} = N_{s_i} - N_{s_j} ,$$

where N_{s_i} and N_{s_j} denote the number of neighbors of the cell under consideration that belong to segment s_i and s_j, respectively. If $\Delta L_{(s_i,s_j)} < 0$, the boundary length would be reduced, and it would be increased if $\Delta L_{(s_i,s_j)} > 0$. Thus, we obtain the desired behavior regarding the change in boundary length if and only if the sign of $\Delta L_{(s_i,s_j)}$ equals the sign of $d_{L,(s_i,s_j)}$. Based on this consideration, we can define the second rule for the cellular automaton:

R3.2 A cell in state s_i should change its state if $d_{L,(s_i,s_j)}\Delta L_{(s_i,s_j)} > 0$ for any state s_j of a cell in its von Neumann neighborhood.

These two rules consider the optimization criteria defined above, but they do not consider the segments' shapes. More compact shapes are generally preferable as they are easier to interpret. Therefore, we also include a so-called security factor introduced by Dorling [84]. The security factor is computed for each cell and measures how exposed the cell is with respect to the other cells belonging to the same segment. To compute the security factor, the number of von Neumann neighbors belonging to the same segment is multiplied by 3, and the number of neighbors on the diagonal that belong to the same segment is added. Therefore, the security factor ranges from 0 (for isolated cells) to 16 for cells surrounded by cells of the same segment. Based on the security factor, we define a third rule:

R3.3 A cell should only change if R3.1 or R3.2 is fulfilled and the security factor lies below a threshold of 11.

Note that the threshold of the security factor chosen here differs from that chosen by Dorling [84]. A detailed investigation of the threshold for the security factor and the reasoning for this choice is presented in Section 3.6.

The previous rules define if a cell should change its state but not yet the state it should be in after the transition. Therefore, we define a fourth rule:

R3.4 A cell in state s_i should change to state s_j if R3.3 is fulfilled and $s_j = \arg\max_{s_k \in S_{vN}}(\max(\hat{d}_{L,(s_i,s_k)}, d_{A,s_k}))$, where $\hat{d}_{L,(s_i,s_k)} = |d_{L,(s_i,s_k)}|$ if $d_{L,(s_i,s_k)} \Delta L_{(s_i,s_k)} > 0$, and $\hat{d}_{L,(s_i,s_k)} = 0$ otherwise. Here, S_{vN} is the set of states in the von Neumann neighborhood of the segment.

The background cells present in the initial configuration should vanish over the iterations of the algorithm. This can be achieved by setting the area deviation of the background to -1, which is the minimum value that the area deviation could take. Thus, any segment would spread in background regions, causing the number of background cells to reduce over time.

3.5.2.3 *Topology Preservation*

While the previously defined rules optimized for area and boundary lengths preservation and take the shape of the segments into account, they do not ensure topology preservation (see task T3.1). Therefore, we follow an adapted version of Dorling's topology preservation methodology [84]: Counting the number of segment changes when going around the cell in its Moore neighborhood (see Figure 3.4) allows for determining the criticality of the cell. A cell is considered critical if changing the state to one of its neighbors violates the topology preservation even though changing to another neighbor's state might preserve it. The cell is critical for preserving the topology if the segment labels change more than 3 times. In this case, the cell should not be changed, independent of rules R3.1 to R3.4. As we add an additional border of vertices with the fixed index -1, we also ensure topology preservation on the boundary. To avoid vanishing segments, isolated cells (security factor of 0)

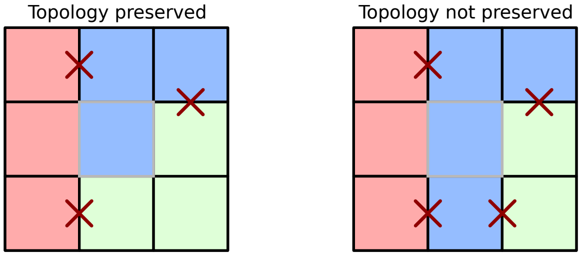


Figure 3.4: The topology of the graph might be violated if the number of segment changes (dark red crosses) when traversing the Moore neighborhood of the central cell (gray border) is > 3. Otherwise, the topology is preserved.

are not allowed to disappear. However, an exception to this rule is the background. The size of the background segments should shrink and, if topology preservation allows for it, vanish entirely. Therefore, background cells with a security factor of 0 are also allowed to vanish.

However, the topology preservation criterion might prohibit the vanishing of background cells. Therefore, the background cells marked as critical are checked separately. One can directly determine if the cell's new state would destroy the topology by checking the new boundaries in the original graph. However, it is not always possible to remove all background cells while preserving the topology. This is, for example, the case for complex, non-planar graphs. In these cases, we keep the background cells as separating boundaries and refer to them as *separators*. To distinguish them from segments, we treat them differently in the visualization (see Section 3.5.3). Cells in a state -2 that indicate edge crossings can never change. Due to the crossing, they contain at least four changes between segments when traversing the surrounding cells.

3.5.2.4 *Iterative Optimization*

Based on the definition of the cellular automaton in the previous sections, the 2D embedding is computed iteratively. However, changing all cells at once might lead to conflicts as neighboring cells can change simultaneously. In this case, the topology preservation, as described in the previous section, would not hold. To tackle this problem, Dorling [84] proposes to use a checkerboard pattern. In this way, the von Neumann neighborhood will be preserved. However, to avoid misinterpretations, we want to preserve the topology in the Moore neighborhood and, thus, avoid diagonal neighborhoods that are not represented by the graph. The checkerboard pattern does not ensure this. Therefore, we apply a

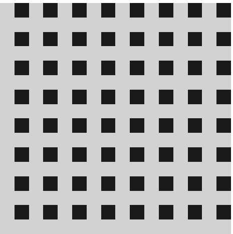


Figure 3.5: We superimpose a pattern to avoid simultaneous changes of adjacent cells, which could violate the topology. When overlying this pattern, only the cells colored in black are changed.

pattern as shown in Figure 3.5 that neither allows direct nor diagonal neighbors to change in the same timestep. In each iteration, the pattern is shifted. In this way, each cell could change its state every fourth iteration.

The user can set the number of iterations. However, the cellular automaton should terminate early based on a convergence criterion. We could consider the algorithm converged if no changes occur any more. Therefore, after each iteration, we check if any cell varied. As each cell could only change every fourth iteration, we require four iterations without changes for the algorithm to converge.

Even though neighboring cells cannot change simultaneously, several cells still change at once, which could lead to an oscillation without further improvement. Therefore, we introduce damping by defining a state's probability of switching. As the damping should get stronger with increased accuracy in representing the segment and boundary sizes, we define the probability as the absolute value of the maximal deviation of either the areas or the boundaries. To improve the results, we introduce a user-defined scaling factor g applied to the damping. Note that a higher value of g indicates a higher tolerance to the deviation and, thus, a weaker damping. To take the stochastic nature of the cellular automaton into account, we increase the threshold for stopping to ten iterations in case the damping is used. However, this threshold might also be adapted by users.

The original node-link diagram layout of the graph might include some unnecessary edge crossings when showing segments in a dense visualization. Two different cases might occur:

1. Two segments could cross multiple times, as shown for the light green and cyan segments in Figure 3.6. When removing one of these crossings, one needs to ensure that the segments are not split.

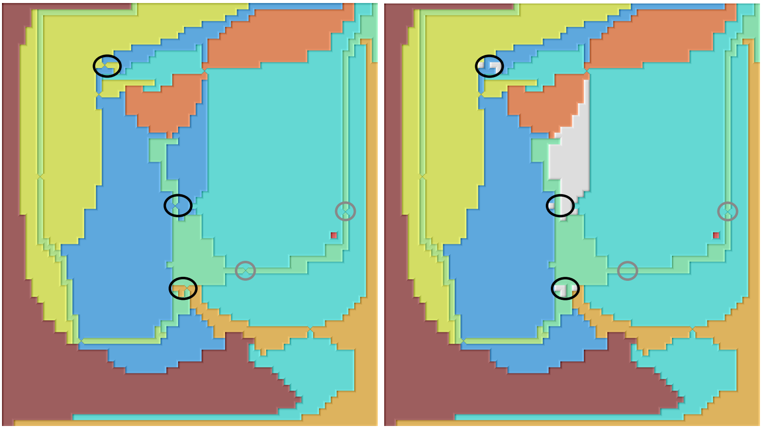
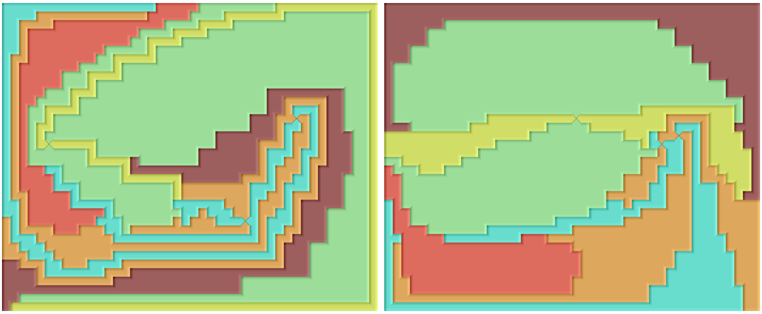

(a) Before. (b) After.

Figure 3.6: Crossings between segments that become unnecessary can be removed to improve the embedding. Crossings can become unnecessary in case of duplicate crossings (gray circles) or if the topology of the segment is already covered by other parts of the segment (black circles).

2. An edge crossing was needed in the graph embedding to ensure a connection to another segment or the boundary. While the background vanishes, it might happen that this connection is also made in other parts of the segment. In this case, the original edge crossing and the corresponding part of the segment can be removed entirely. This case is shown with black circles in Figure 3.6. When removing the part of the segment and the edge crossing, the state of the corresponding cells is set to 0, corresponding to the background cells. Thus, we obtain additional space for the other segments to spread further.

However, testing the conditions for removal is computationally expensive. Therefore, we only perform it in a user-defined interval which we choose as every 300th iteration for the results presented here.

Without further modifications, the positions of the edge crossings are fixed. However, this might lead to unnecessary complex layouts, as shown in Figure 3.7a. Figure 3.7b shows that the layout significantly improves if the edge crossings could be moved as the segments are more compact and show less line-like structures. For obtaining more compact segments, we aim at moving the crossings towards the barycenters of the segments. The crossings are moved by switching the state of the cell in state -2 to the state of the neighboring cell. Before performing the



(a) Fixed crossing positions.

(b) Flexible crossing positions.

Figure 3.7: Fixed positions of the crossings might lead to unnecessary complex layouts (a). By moving the crossings towards the barycenter of the segment, one obtains more compact segments (b). Both examples were created by applying the cellular automaton on the same initialization.

movement, it is checked if the topology is preserved. Thus, the crossings are only moved if a horizontal or vertical movement is possible without further changes.

3.5.3 *Segmentation Visualization*

Segmentations in 2D are commonly visualized by assigning different colors to the different segments. We follow a similar approach but aim for a color map-independent visualization. The colors may be chosen based on a color map and optimized to, for example, show distinguishable colors among neighboring segments. However, color can also be used to encode further information like values assigned to the segments in the multi-dimensional space or to encode the area deviations of the embedding to visualize embedding artifacts. As the latter cases might lead to similar colors of segments with a joint boundary, we want to emphasize the segment boundaries independent of the concretely chosen color coding.

For visualizing the segment boundaries, we apply a shading approach inspired by cushion treemaps [352, 336], which provides a large flexibility in the choice of color. We further aim at adapting the shading to visually differentiate the separators (regions of background cells that remain to separate different segments) from the actual segments. As our segmentations are not hierarchical and we aim at a relatively simple visualization without additional clutter, we want to keep the inner part of the segment flat. Like cushion treemaps, we use quadratic functions, but

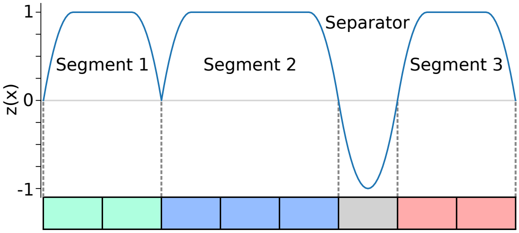


Figure 3.8: The height profile for the shading applied to the segmentation should contain plateaus in the central parts of the segments. Separators are modeled as valleys to distinguish them visually. For this graph, we set the height $h = 1$ and the width for height changes $w = 0.5r$ where r is the number of pixels used for each cellular automaton cell.

for achieving flat regions of varying size, we use piecewise definitions of the height function as shown for the 1D case in Figure 3.8. The height function $z_i(x)$ uses an increase of the height for the segments but a decrease for the separators. We color the separators clearly distinguishable from the color map applied to the segments for additional differentiation. In all examples presented in this chapter, we chose white for the separators, as white was absent in any color maps used.

As x and y directions are treated identically, we describe our shading based on the x-direction of the automaton. In the following, we can also treat each cell of the cellular automaton as a superpixel of an image. For defining the height function $z_i(x)$ we apply the constraints

$$\begin{aligned} z_i(x_{i,1}) &= 0, \\ z_i(x_{i,1}+w) &= h, \qquad (3.3) \\ \frac{dz_i}{dx}(x_{i,1}+w) &= 0. \qquad (3.4) \end{aligned}$$

The position where segment s_i starts is denoted as $x_{i,1}$, w is the width of the region where the function increases quadratically, and h is the height of the central parts of the segments. The constraints for the decrease at the end of the segment are analogous.

The separators are modeled as valleys with a negative height, as shown in Figure 3.8. This results in the following constraints:

$$\begin{aligned} z_v(x_{v,2}-w) &= -h, \\ \frac{dz_v}{dx}(x_{v,2}-w) &= 0, \qquad (3.5) \\ \frac{dz_v}{dx}(x_{v,2}) &= \frac{dz_i}{dx}(x_{i,1}). \qquad (3.6) \end{aligned}$$

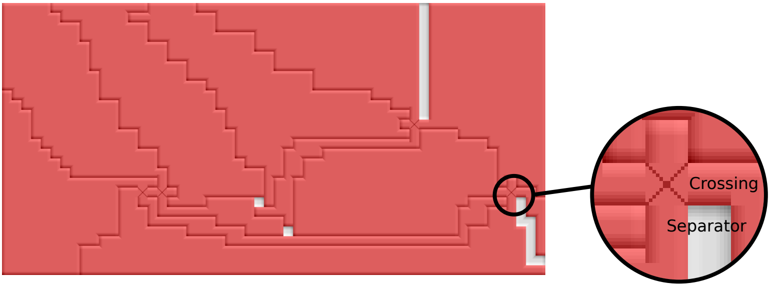


Figure 3.9: The shading of segments, separators, and edge crossings allows for visually distinguishing them, even independent of the color coding.

The separator v ends at position $x_{v,2}$. The constraints for the start of the separator are analogous. These constraints lead to the coefficients a, b and c for the segment or separator ranging from x_1 to x_2 in x-direction and from y_1 to y_2 in y-direction:

$$a = \begin{cases} 0 & x_1 + w < x < x_2 - w \\ -\frac{\gamma h}{w^2} & \text{otherwise} \end{cases},$$

$$b = \begin{cases} 2\frac{\gamma h}{w^2}(x_1 + w) & x_1 \leqslant x \leqslant x_1 + w \\ 0 & x_1 + w < x < x_2 - w \\ 2\frac{\gamma h}{w^2}(x_2 - w) & x_2 - w \leqslant x \leqslant x_2 \end{cases},$$

$$c = \begin{cases} 2\frac{\gamma h}{w^2}(y_1 + w) & y_1 \leqslant y \leqslant y_1 + w \\ 0 & y_1 + w < y < y_2 - w \\ 2\frac{\gamma h}{w^2}(y_2 - w) & y_2 - w \leqslant y \leqslant y_2 \end{cases},$$

where γ is a parameter used to differentiate between segments ($\gamma = 1$) and separators ($\gamma = -1$). These coefficients can then be used to compute the normals for the shading as $(2ax + b, 2ay + c, 1)$.

In this way, we can visualize segments as well as separators. We also want an intuitive visual encoding for showing the edge crossings. It should be clear that there is a crossing and should not be confused with boundaries. Additionally, it should not pop out too much, as the crossings should not be highlighted. Based on these considerations, we decide to use a visual encoding that resembles a cross as shown in Figure 3.9 where we can see that this visualization is also applicable in case all segments show the same color. This is achieved using shading on the diagonals where we set the normal vector to $(0.1, 0.1, 1)$. When applying color coding with different colors, the segments are colored according to the color of the adjacent segment. Figure 3.9 shows that the shading of the segment boundaries also allows for clearly separating the boundaries independent of the color coding.

3.6 ALGORITHMIC EVALUATION

To study the influence of the different optimization criteria as well as the scalability of the algorithm, we perform an algorithmic evaluation based on synthetic datasets. Here we opt for synthetic datasets as they allow us to adapt the properties of the data, like the dimensionality and the number of segments, based on our needs.

We define a dataset D_1 that should allow for creating segmentations with a defined number of segments in varying shapes and a user-defined resolution. Therefore, random seed points are used to grow regions with a random growth rate. Here, each unassigned cell is added to one of the segments present in its neighboring cells based on the growth rate as a probability. With this methodology, we create 2D and 3D segmentations. 2D embeddings of 2D segmentations are no meaningful use case. However, as they allow a simple visual comparison, they provide a good intuition on how our algorithm works.

The second dataset, D_2, is based on a 3D cube divided once in all dimensions. Thus, it contains eight equally sized segments. Each segment has a joint boundary with three other segments and the outer boundary. Figure 3.2 in the top-left shows this dataset as a volume rendering.

3.6.1 *Quality Criteria*

For the algorithmic evaluation of the approach, the following set of quality metrics is used:

- *Number of edge crossings:* Fewer edge crossings mean less visual complexity of the visualization. Additionally, edge crossings are a projection artifact that should be minimized. Therefore, we investigate the number of edge crossings that are present in the embedding.

- *Mean area deviation:* The area preservation quality can be determined by computing the area deviation for each segment. As a global quality metric, the mean area deviation $\bar{d_A} = \frac{1}{|S|} \sum_{s \in S} |d_{A,s}|$

is used. The area deviation $d_{A,s}$ is computed by Equation 3.1 and $|S|$ denotes the number of segments in S.

- *Mean boundary length deviation:* The mean boundary length deviation allows for assessing how well the boundary length optimization works. It is computed as $\bar{d_L} = \frac{1}{|E|} \sum_{(s_i,s_j) \in E} |d_{L,(s_i,s_j)}|$ where s_i and s_j are adjacent segments and $d_{L,(s_i,s_j)}$ is the boundary length deviation defined in Equation 3.2. The number of edges in the graph is denoted as $|E|$.

Note that we ensure topology preservation and, thus, we do not introduce a quality criterium to measure the topology preservation. It is verified that the topology is correctly preserved when executing the algorithm by comparing the graph of the embedding with the graph of the multi-dimensional segmentation, and we found that our algorithm works correctly.

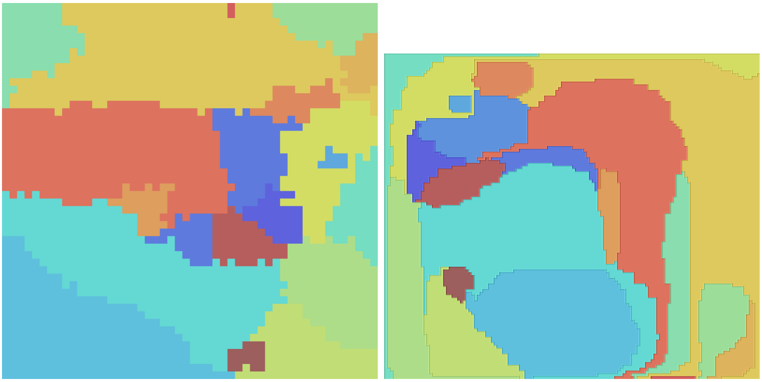
3.6.2 *Results*

In the following, we discuss the evaluation results mainly based on the synthetic datasets and considering the quality metrics.

3.6.2.1 *Visual Comparison*

We apply our algorithm to a 2D segmentation of dataset D_1 that contains 20 segments to verify our algorithm. Thus, the input data also provides a ground truth even though the positions and shapes of the embedding are allowed to vary. The input data and the embedding result are shown in Figure 3.10. The embedding result was computed using $5,000$ iterations, a scaling factor for the damping of $g = 7$, and a security factor threshold of 11. The topological structure of the segmentation is preserved. A visual comparison reveals similar segment sizes. The mean area deviation of $\bar{d_A} = 0.017\%$ confirms this observation.

Also, the boundary lengths are perceived as similar, for example, when looking at the brown segment in the lower left of Figure 3.10b. Most of its boundary is shared with the green and the lighter blue segment, while only a short part is shared with the segment in darker blue. These observations agree well with the boundaries in the original data (see Figure 3.10a). The similarity in boundary lengths over all segments is confirmed by the mean boundary length deviation of $\bar{d_L} = 0.96\%$. However, individual segments might deviate. Nevertheless, in general, the boundary lengths are well preserved. When visually comparing the segmentations, it is obvious that shapes and locations are not preserved.



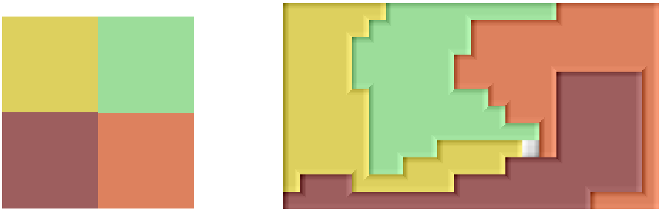
(a) Original.

(b) Embedding.

Figure 3.10: A 2D segmentation (a) created as part of dataset D_1 can be visually compared to its 2D embedding (b). The topology is preserved, including those of the light blue segment (on the right in a), top-left in b)) surrounded by the yellow one. Area sizes and boundary lengths are also approximately the same, while the shapes and locations of the segments differ.

However, this is not the goal of this algorithm, as these aspects do not apply to higher-dimensional inputs, which are the target of our algorithm.

The results for applying our approach to dataset D_2, including intermediate steps, are shown in Figure 3.2. The segments appear to have a similar size confirmed by the mean deviation of $\bar{d_A} = 0.089\%$ and a mean boundary length deviation of $\bar{d_L} = 5.52\%$. The dimensionality re-



(a) Original.

(b) Embedded.

Figure 3.11: A 2D segmentation (a) with four equally sized segments cannot be embedded (b) by our algorithm without introducing separators because the diagonal neighborhood in the center is not represented by the graph.

Table 3.1: Quality criteria for different real-world datasets with dimension n and number of segments $|S|$. Crossings denote the number of edge crossings after finishing the algorithm, and resolution provides the image resolution after finishing the cellular automaton.

Dataset	n	$\|S\|$	**Crossings**	**Resolution**	$\bar{d}_A$ (%)	$\bar{d}_L$ (%)
D_1	2	20	0	116 × 134	0.017	0.963
D_2	3	8	4	46 × 28	0.089	5.517
Ablation	3	59	14	374 × 448	0.547	1.708
Nucleon	3	42	17	208 × 270	3.138	1.766
Synthetic	4	4	0	18 × 18	1.299	8.264
Blood flow	5	4	0	18 × 18	0.170	0.495
Semiconductor	4	13	13	114 × 128	1.576	4.045

duction can explain the significantly higher deviations compared to the previous case. Additionally, the graph of the chosen dataset is nonplanar, resulting in crossings in the segmentation embedding. This results in four edge crossings. Additionally, some separators are located between segments without a joint boundary. They are also unavoidable, which can be easily seen by our neighborhood definition. In the 3D dataset D_2, the different segments all come together on the diagonal in the center of the domain. However, as we do not include diagonal neighborhoods in our algorithm, this behavior cannot be covered accurately. This can also be shown in a 2D example as presented in Figure 3.11. Even in this 2D case, a dense visualization without separators is not possible if only the von Neumann neighborhood is considered for creating the graph. This is caused by the topology preservation of the cellular automaton which prevents diagonal neighbors.

In 2D, this leads to a relatively small loss of neighborhood information. However, for higher-dimensional data, the neighborhood structure of von Neumann neighbors is very sparse as, by definition, only the neighbors along a single dimension are considered [394]. For the applications presented in this work, this does not impose problems since the dimensionality of none of the presented examples is very high. Additionally, the neighborhood criterium for creating the graph can be easily exchanged if another one is more suitable for the application scenario.

3.6.2.2 *Performance for Different Datasets*

Table 3.1 shows the numerical evaluation for different datasets. In this section, we provide an overview of the applicability of our algorithm to different datasets, while the influence of the different factors is provided in the following sections. The timings for these datasets are discussed in more detail in Section 3.6.2.6. All results were created with a scaling factor for the damping of 7 and a security factor threshold of 11. If not stated otherwise, $5,000$ iterations were used for the cellular automaton.

The datasets abbreviated as D_1, and D_2 in Table 3.1 refer to the datasets discussed in Section 3.6.2.1. The datasets *Ablation* and *Nucleon* are 3D segmentations that we will discuss in more detail in Section 3.7. The datasets called *Synthetic*, *Blood flow*, and *Semiconductor* refer to the parameter spaces of the simulation ensembles investigated in Chapter 4. We do not discuss the datasets in more detail here, as in this point, only the accuracy based on the characteristics, such as the number of segments, is investigated.

In general, the accuracy strongly depends on the dataset. The blood flow dataset converged after 1134 iterations. The relatively large deviations, for example, when compared to the dataset D_1, can be explained by the small resolution of the output image, which does not permit a higher accuracy. The synthetic parameter-space partitioning shows very high deviations in the size of the segments and the boundaries. When investigating the result in more detail, one can identify that one of the segments has a very high boundary deviation. Due to the small number of segments, it strongly influences the accuracy of the result.

3.6.2.3 *Influence of Parameters*

Different parameters drive the algorithm presented in this chapter. Two key parameters are the choice of the scaling factor for the damping g and the threshold of the security factor. To understand their influence, we use dataset D_1 with 5, 10, and 15 segments covering 2D and 3D domains. We study scaling factors for the damping between 1 and 9 and security factors between 9 and 12 where 12 is the value initially proposed by Dorling [84]. As the resolution of the cellular automaton scales with the complexity of the segmentation, the number of iterations of the cellular automaton is heuristically set to the number of cells.

The results for the area and boundary length deviation in dependency on the input parameters are shown in Figure 3.12. The accuracy slightly increases for $g > 3$. This is expected as strong damping, corresponding to small values of g, could cause the algorithm to terminate early before

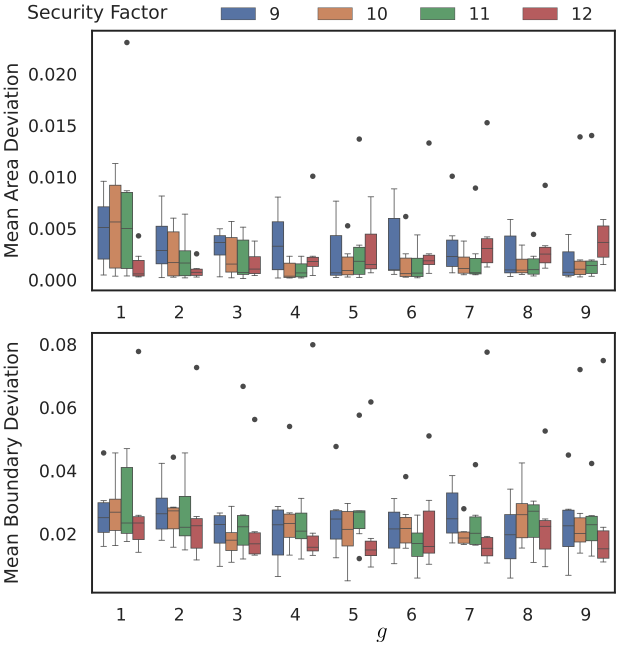
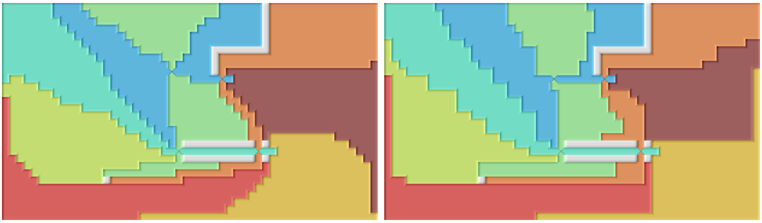


Figure 3.12: Mean area deviation and mean boundary length deviation are computed based on the scaling factor for the damping g and the security factor to understand the influence of the parameters.

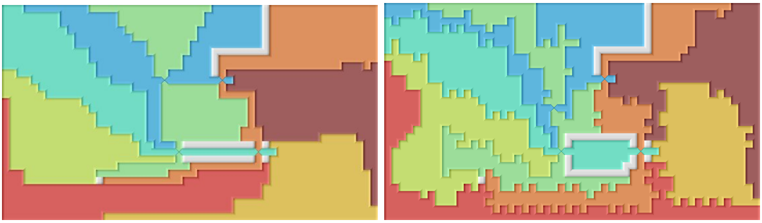
a minimum in the optimization can be reached. For larger scaling factors for the damping, no significant variation is visible.

For the security factor, we observe the best results for thresholds of 10 and 11, especially with respect to the mean area deviation. When observing the mean boundary length deviation, a security factor of 12 seems to produce the best results at the cost of a larger area deviation. This observation can be explained by the larger variability in boundary shape allowed for higher security factor thresholds. This also becomes evident when visualizing the results for different security factors, as shown in Figure 3.13. Observing the four different results for the different security factors reveals negligible differences between security factor thresholds from 9 to 11. A security factor threshold of 12, however, leads to more granular boundaries between the segments. Even though it slightly increases the mean boundary length deviation, it adds significantly more complexity and decreases the mean area deviation. Therefore, we recommend choosing a smaller security factor threshold.



(a) Security factor 9.

(b) Security factor 10.



(c) Security factor 11.

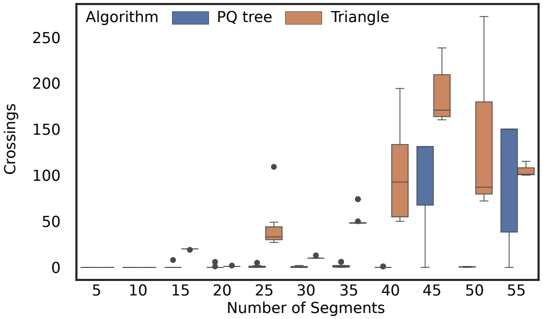
(d) Security factor 12.

Figure 3.13: The embeddings with a security factor of 9 to 11 differ slightly, while the embedding for a security factor of 12 shows more complex boundaries between the segments. All embeddings were created using the same initial configuration.

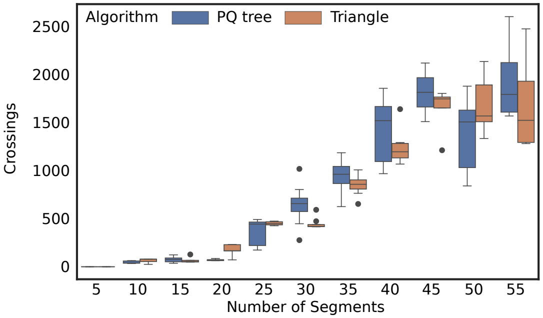
Based on the numerical results, when considering both optimization criteria and the visual investigation, we recommend a security factor of 10 or 11. We opt for 11 in the remainder of this chapter.

3.6.2.4 *Influence of Graph Embedding*

As the graph embedding is the basis for all other steps of our algorithm, it strongly influences the final visualization. Note that the chosen graph algorithm can be easily exchanged. As graph drawing is a very complex topic, going into detail is beyond the scope of this work. However, to evaluate the influence, we want to evaluate the number of edge crossings based on the computation of the maximum planar subgraph. Additionally, we capture the variation over the choice of the external face. We choose dataset D_1 with 5 to 55 segments in steps of 5. We consider both 2D and 3D input data. We compute all the graph drawings for all possible external faces to consider the variation over the choice of drawings. For the final evaluation, we only consider successful drawings. Regarding the choice of the planar subgraph computation, we use the PQ tree-based algorithm and the triangle-based algorithm as introduced in Section 3.2.1.1.



(a) 2D.



(b) 3D.

Figure 3.14: The number of edge crossings varies depending on the choice of the subgraph algorithm and the choice of the external face. For 2D input data (a), that results in planar graphs, only the PQ tree-based subgraph algorithm finds at least one embedding without edge crossings. For 3D segmentations, none of the two tested algorithms is clearly preferable. However, in all cases, the choice of the embedding with the smallest number of edge crossings significantly improves the result.

The results for 2D inputs are shown in Figure 3.14a. For 2D segmentation data, the representing graph must be planar by definition. However, while the PQ tree-based algorithm always finds at least one planar embedding, this does not apply to the triangle-based algorithm. The wide range of edge crossings that might be present in the graph drawing shows that computing different graph embeddings and choosing the best one with respect to edge crossings significantly improves the result.

The evaluation for 3D data is presented in Figure 3.14b. We observe better results for the triangle-based algorithm in some cases, for example, for 45 or 55 segments, while in other cases, like for 50 segments, the PQ

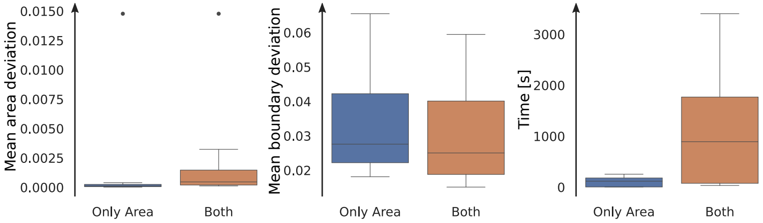


Figure 3.15: Comparing only area optimization to area and boundary optimization shows the tradeoffs between both criteria. While optimizing for both increases the area deviation and, even more significantly, the computation time, the boundary length deviation decreases.

tree-based algorithm is better. However, the number of edge crossings is generally in the same order of magnitude, so we cannot recommend one algorithm over the other. In the examples presented in this chapter, we used the PQ tree-based algorithm because it performed better for planar graphs, which could also occur for higher-dimensional inputs. Nevertheless, if prior knowledge about the graph representing the segmentation is available, the algorithm can be easily exchanged.

In some of the investigated cases, a more sophisticated edge routing algorithm for the edges connecting the vertices representing segments to the boundary would further decrease the number of edge crossings. However, choosing the best graph drawing among different ones yields sufficiently good results. Additional postprocessing to remove unnecessary edge crossings in later steps could also remove some of the edge crossings introduced by the simple edge rerouting.

3.6.2.5 *Optimization Criteria*

The influence of the different optimization criteria, namely area and boundary length preservation, should also be evaluated. As the boundary length optimization does not lead to spreading the segments and, thus, to reducing background voxels, we do not consider only optimizing for boundary length. Instead, we compare using only the area optimization to a mutual optimization of area and boundary length. For this evaluation, we use dataset D_1 with 3D segmentation of 5 to 15 segments in steps of 1. As the area deviations are usually smaller than the boundary length deviations, considering only area deviations with a scaling factor for the damping of 7 could lead to early stopping. Therefore, we increase it to 100 when excluding the optimization of the boundary length.

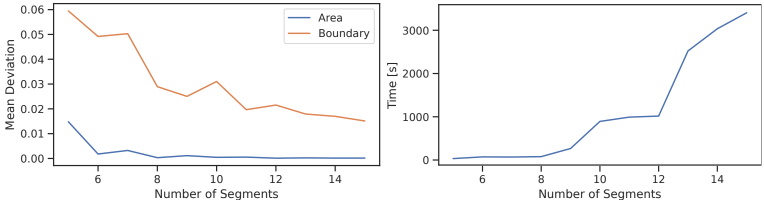
The results for the different optimization criteria and computation times are shown in Figure 3.15. Adding the boundary optimization leads to a larger area deviation and a smaller boundary length deviation. This meets our expectations as we obtain a trade-off for both optimization criteria. However, it is notable that the difference, especially in the boundary length deviation, is relatively small. When considering the computation times, the difference is larger, as optimizing only for area preservation is significantly faster. Thus, we recommend choosing only the area preservation if faster computation times are more important than boundary length preservation. For example, this might occur for segmentations represented by large and complex graphs. If area preservation is mainly of interest, one might also consider excluding the boundary length preservation to obtain better results. We consider both for the remainder of the chapter, as we want to include both aspects in the analysis.

3.6.2.6 *Scalability*

For the last algorithmic analysis, we want to consider the scalability of our approach, which depends on different factors based on the different steps. The algorithm starts with the graph computation. The vertices and their weights can be computed in $\mathcal{O}(N)$ for N grid cells in the multi-dimensional input domain. For n-dimensional input data, the shared boundaries can be determined in $\mathcal{O}(Nn)$ because both neighbors in each dimension must be checked for each grid cell. However, the number of grid cells usually grows with an increasing dimension. When assuming the same sampling, the number of grid cells grows exponentially.

All the following steps only depend on the weighted graph and are, thus, independent of the number of samples and the dimensionality. Instead, the computational complexity depends on the internal complexity of the segmentation that should be visualized. The number of vertices and edges can characterize the complexity of the graph. With the increasing complexity of the graph, not only do the computational costs for the graph drawing increase but also the number of cells in the cellular automaton that are required to draw the graph. This leads to increased computation times for the iterative computation to optimize areas and boundary length. Edge crossings further increase computation times as additional steps for their movement and potential removal need to be executed.

The mean area and mean boundary length deviation in dependence on the number of segments are shown in Figure 3.16a. In general, we observe a decrease in the deviations. However, this is also caused by an



(a) Mean Deviation. (b) Time.

Figure 3.16: The mean deviations decrease for both area and boundary size, with an increase in the number of segments (a). At the same time, the increased graph complexity leads to an increase in the computation times (b).

Table 3.2: Timings for the individual steps of different datasets with N input voxels and M pixels in the output.

Dataset	N	M	**Graph (s)**	**Embedding (s)**	**Automaton (s)**
D_1	2,500	15,544	0.006	0.576	244.122
D_2	8,000	1,568	0.037	0.353	45.706
Ablation	778,688	166,656	2.851	129.784	4809.0
Nucleon	68,921	56,160	0.245	1.669	1112.2
Synthetic	10,000	324	0.086	0.008	14.086
Blood flow	3,200,000	324	26.446	0.009	3.294
Semiconductor	10,000	14,592	0.090	5.562	257.648

increasing number of cells available for the cellular automaton. Thus, if the deviations are too high using our standard algorithm, one option to improve the values might be increasing the available cells. This can be achieved by removing the simplification step where duplicate rows and columns are removed. If the number of cells should be further increased, the multiplication factor f for the resolution could be increased manually. However, increasing the resolution comes at the cost of significantly larger computation times.

Figure 3.16b shows the timings for dataset D_1 with 5 to 15 segments for 3D data. All timings were obtained on a laptop with a 1.7GHz AMD Ryzen Pro 7 processor. The computation times increase significantly with the number of segments, which can be explained by the more complex graph needing more grid cells and more edge crossings.

The computation times of the different steps vary strongly among the different datasets. Therefore, we provide an overview of the computation times of each dataset where we use the datasets discussed in Section 3.6.2.2. The results are presented in Table 3.2.

The most time-consuming step depends on the dataset. In general, the number of voxels in the input space determines the computation time for the graph, which confirms our theoretical considerations. The time for the embedding depends on the complexity of the graph for which the number of segments and the number of edge crossings (see Table 3.1) provide an indicator. Finally, the computational cost for the automaton is mainly dominated by the number of automaton cells which equals the number of pixels in the output. The computation times for the cellular automaton for the blood flow dataset are significantly lower than those of the parameter-space partitioning of the synthetic dataset. This is caused by the early termination of the optimization for the blood flow dataset. The longest computation times for applying the cellular automaton are found for the ablation and nucleon datasets. However, these two datasets also need a significantly higher resolution for the cellular automaton.

3.7 EMBEDDING OF 3D SEGMENTATIONS

To demonstrate the applicability of the approach, we apply it to 3D image segmentations. The application to multi-dimensional parameter-space partitionings is discussed in Chapter 4. Another application scenario for 3D segmentations of ensemble data is presented in Chapter 8.

As a first application, we study the 3D volume segmentation used to create the radiofrequency ablation simulation that will be analyzed in more detail in Chapter 7. The segmentation contains 11 labels that indicate different organ tags and tags for the tumor and the needle. Just as the simulation output, the segmentation has a resolution of $92 \times 92 \times 92$. However, the regions with unique organ tags are not connected. Therefore, we set a unique ID for each connected component resulting in 59 segments.

Figure 3.17 shows the result when applying our algorithm. For the color coding, we apply a color map on the original organ tags such that different segments with the same tag are shown in the same color. When observing the embedding, it becomes directly evident that the two largest segments are the liver and a region without organ tags available. Those two segments cover the majority of the 3D domain.

In the center of the embedding, a group of segments with common boundaries represents the vessels. One can observe that the portal vein (PV) and the hepatic artery (HA) share a boundary with the unlabeled tissue, while the hepatic vein (HV) is not connected to the correspond-

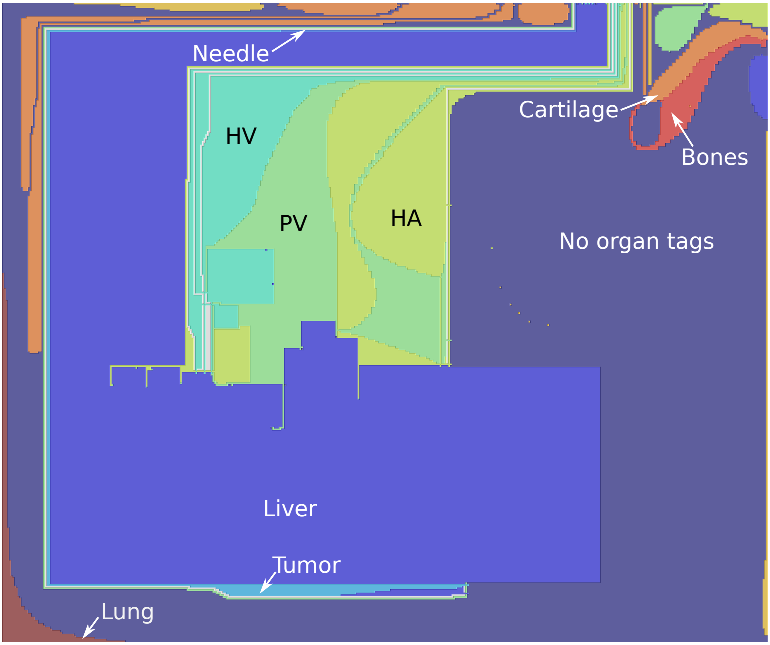
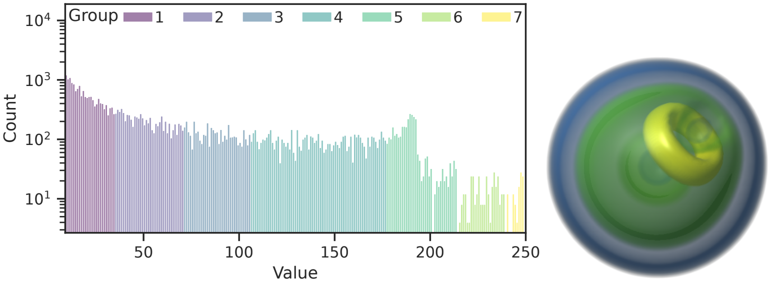


Figure 3.17: The embedding of the 3D image segmentation used for the ablation dataset is color-coded according to the organ tags. The visualization reveals information about the sizes of the different regions as well as about the topological structure of the segmentation.

ing segments. We can spot tiny segments belonging to the hepatic artery inside the large segment representing tissue without organ tags. These segments are completely enclosed by the other segment. However, artery parts disconnected from the boundary and other veins are not plausible. Therefore, these segments indicate either mistakes in the segmentation or undersampling artifacts. When investigating the embedding in more detail, we can identify that the needle used for the ablation is only connected to liver tissue, the tumor, and unlabeled tissue. Based on this observation, one can state that the needle placement does not cause damage in other tissue types.

As a second application example, we use the nucleon dataset with a resolution of $41 \times 41 \times 41$. We partition the dataset based on the histogram as shown in Figure 3.18a and then identify connected components. This leads to 42 segments. Three selected groups based on the histogram are shown in Figure 3.18b. For computing the embedding, we



(a) Histogram.

(b) Volume visualization.

Figure 3.18: The nucleon dataset can be partitioned based on its histogram (a). We can only show a subset of the regions in volume visualization (b).

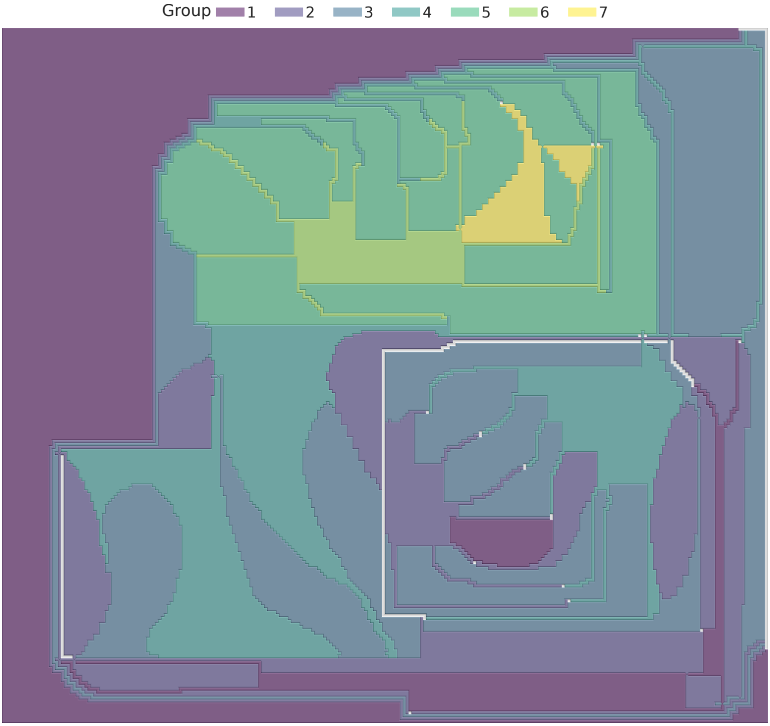


Figure 3.19: The 2D embedding of the partitioned nucleon is color-coded according to the segments defined in Figure 3.18a.

used $2,000$ iterations. The resulting embedding is shown in Figure 3.19, where we used the group IDs as shown in the histogram in Figure 3.18a for the color coding. The embedding allows for investigating the structure of the segmentation. It can be observed that the violet region (label 1) not only appears as surrounding the nucleon and is the only segment connected to the boundary, but it also appears in a separate segment inside. The yellow (label 7) and light green (label 6) regions are surrounded by the region labeled with 5. However, both regions connect to those labeled as 4. Additionally, regions corresponding to labels 3 and 5 are split into several segments. Here, one can deduce that we again observe artifacts caused by a relatively coarse sampling.

3.8 DISCUSSION

In this chapter, we presented an algorithm to compute 2D embeddings of multi-dimensional partitionings or segmentations. In the embedding, we preserve the topology while optimizing the area and boundary sizes according to the original areas and boundary sizes in the multi-dimensional space. To create the embedding, a graph is derived from the multi-dimensional segmentation. A drawing of the graph provides the initial configuration for a cellular automaton that optimizes the areas and boundary sizes. Finally, we apply shading to visualize the segments while allowing for various color codings.

Besides fulfilling the optimization criteria, our approach allows for identifying structures like segments enclosed by other segments or connections to the boundary. The algorithmic properties were evaluated based on synthetic datasets. At the same time, the application to 3D data provided an overview of the segmentation that cannot be obtained in other visualization approaches like volume renderings without a large amount of interaction and colormap optimizations. The embedding becomes even more helpful for higher-dimensional spaces, as shown in the following chapter.

An alternative visualization of the multi-dimensional segmentation can be obtained by interpreting the grid points as samples and projecting the samples using common dimensionality reduction techniques like MDS or UMAP. However, these techniques do not preserve the topology and cause many overlaps between the different segments. While overlaps might be solved using a variant of approaches for decluttering scatterplots [72], the topology preservation is not ensured. Another option to reduce the dimensionality might be using space-filling curves to obtain space-filling visualizations [399]. However, also in this case, the topology is not preserved. Therefore, these visualizations also cannot be evaluated with respect to the segment and boundary size preservation. For a direct comparison of our segmentation embedding to dimensionality reduction, see Chapter 4. We show embeddings of the parameter-space samples together with the respective segmentation embedding and discuss the advantages and disadvantages in more detail.

The node-link diagram that we use as a starting point for the algorithm already contains the full topological information. There are also alternative approaches to incorporating the segment and boundary sizes in the node-link diagram, for example, using node sizes and edge widths. While our approach uses the space efficiently, node-link diagrams are a familiar representation to many users. Therefore, in the future, we plan to conduct a user study to better understand the tasks for which our visualization is preferable over a node-link diagram and vice versa.

While the embedding provides an overview of the segmentation, it does not encode the positions and shapes of the segments. For this purpose, the 2D embedding could be linked to other coordinated views, such as volume or surface renderings in the case of 3D segmentations. In these visualizations, selected segments can be shown in their original domain. Thus, we do not consider our visualization a replacement for standard visualization techniques, as they are, for example, common in the medical domain. Instead, our overview embedding focuses on investigating the structure of the segmentation and aims at analysis settings where detailed information on all segments is required.

In its current implementation, the algorithm is computationally relatively expensive. This remains feasible as the computation of the 2D embedding could be performed as a preprocessing step in any interactive setting. However, optimizing and parallelizing the algorithm can significantly improve computation times. In the algorithmic evaluation performed in this chapter, we found that preserving the boundary length was more difficult than preserving the area. Therefore, we plan to improve boundary length preservation in future research. Additionally, in some cases, the segments are connected by rather thin connections which is partially caused by the global optimization of the algorithm and makes the interpretation more difficult. This shortcoming can be improved by including additional optimizations to the cellular automaton, whose performance already significantly improved by allowing the edge crossings to move.

4

ANALYSIS OF PARTITIONINGS

One key goal in the analysis of simulation ensembles is analyzing the parameter space and its effect on to the simulation outcome. Each input parameter can be seen as one dimension forming the multi-dimensional parameter space. Every simulation run is driven by a parameter setting which can be interpreted as multi-dimensional point in the parameter space. However, multi-dimensional parameter spaces, especially with dimensionalities larger than three, are challenging to analyze.

To understand how the parameters influence the simulation outcome, it is often desired to segment or partition the parameters space such that each segment represents parameter settings leading to similar simulation outcomes. In the field of flow simulations, for example, one commonly differentiates between laminar and turbulent flows, which would also partition the parameter space into two distinct regions. Then, one wants to analyze which parameter values lead to which kind of behavior and where the boundaries between both parameter-space segments are located. This corresponds to studying the extent of the segments and the locations of the segment boundaries. While the embedding presented in Chapter 3 allows for showing the topological structure of the partitioning and provides an overview, it does not allow for investigating the extent and locations of the different partitions.

This chapter presents an interactive visual analysis approach that targets the definition and analysis of parameter-space partitions. We start by defining the problem (see Section 4.2) followed by an overview of our approach (see Section 4.3). In the first step of the analysis, a partitioning is created in the similarity space based on the simulation outcome described in Section 4.4. When using the term similarity space, we refer to the space formed by pairwise similarities between the simulation output of the different runs. Second, the similarity space partitioning induces a partitioning on the parameter space. For the analysis of the parameter space, it should be possible to see the shapes of the segments and see the parameter values directly as discussed in Section 4.5. For this purpose, we introduce the hyper-slicer for distortion-free visualization of parameter-space segmentations. Finally, the partitioning should be modified based on the previously obtained insights leading to an interactive visual analysis workflow as presented in Section 4.7. We compare the extended hyper-slicer that is the core of our analysis approach to other distortion-free multi-dimensional data visualization techniques in Section 4.6. Finally, we evaluate our approach by applying it to three real-world simulation ensembles (see Section 4.8).

The results presented in this chapter are mainly based on the following publication:

> **M. Evers** and L. Linsen, Multi-dimensional Parameter-space Partitioning of Spatio-temporal Simulation Ensembles, *Computers & Graphics* 104: 140-151 (2022)

Both authors contributed to discussing ideas, writing, and editing the manuscript. I implemented the approach and created the results.

4.1 RELATED WORK

For a general overview of related work in parameter-space analysis, we refer to Section 2.2.1. However, none of the presented approaches allow for a global, distortion-free overview of the parameter space. Besides these visualizations, methods for navigating through multi-dimensional spaces have been proposed [18, 95]. ManyLands [12] analyzes 4D trajectories based on subspaces in different dimensions, but these approaches are not directly applicable to our analysis tasks.

In this work, we extend the concept of hyper-slices [353], which shows a slice around a focus point for each individual dimension. Thus, hyper-slices can be seen as a variant of scatterplot matrices for continuous data. Hypersliceplorer [342] extends this concept to show multiple slices simultaneously, quickly leading to cluttered visualizations. Therefore, it is not well suited for our purpose of visualizing several segments and their boundaries. HyperMoVal [266] aims at validating regression models and uses hyperslices to visualize the model space. In contrast to the work presented in this chapter, they do not analyze parameter-space partitionings and their properties. Collaris et al. [66] investigate explanations in machine learning models and use hyper-slices to visualize local context in multi-dimensional spaces but do not target simulation ensembles.

	T1	T2	T3	T4
Berger et al. [33]				✗
Fofonov et al. [111]				✗
Luboschik et al. [221]				✗
Matkovic et al. [224]				✗
Piringer et al. [266]				✗
Splechtna et al. [321]				✗
Torsney-Weir et al. [343]				✗
Unger et al. [346]				✗
Obermaier et al. [247]	✗	✗		✗
Wang et al. [361]	✗	✗		✗
Bergner et al. [34]	(✗)	✗	✗	✗
Our approach	✗	✗	✗	✗

Table 4.1: Overview of tasks supported by different parameter-space analysis approaches. Bergner et al. [34] only allow for manual partitioning in two groups and, thus, support (T1) partially.

4.2 TASK DEFINITION

In the context of this chapter, we consider spatio-temporal simulation ensembles as introduced in Section 1.1 but only consider single-field simulations. Additionally, we focus on the analysis of input parameters. While initial conditions might be varied over the ensemble, we do not analyze their influence, which might complicate the analysis using the approach presented in this chapter. Additionally, we require a unique set of input parameters for each simulation run to define the parameter space partitionings. For parameter spaces of dimensionality less or equal to three, direct visualizations using heatmaps for 2D data (see Chapter 5) or volume renderings are applicable. Here, we focus on visualizing parameter spaces with a dimensionality larger than three. However, we assume that the dimensionality does not become too large because, for high-dimensional parameter spaces, dense sampling for spatio-temporal simulation data is computationally infeasible. Thus, the analysis tasks for sparsely sampled, high-dimensional parameter spaces are very different and require further research.

The visual analysis should address the following tasks:

T4.1 *Partitioning* the parameter space. Clusters in similarity space contain a set of simulation runs with similar behavior and induce a partitioning in parameter space. The users should be able to create the parameter-space partitioning by defining the clusters in the similarity space.

T4.2 Creating a *parameter-space overview*. The users should be able to obtain an overview of the parameter-space partitioning such that all partitions can be seen at once. This supports the user in understanding the structure of the parameter-space partitioning and allows for easier navigation in more detailed views.

T4.3 Analyzing *uncertainty and extent* of the partitions. It should be possible to analyze the positions, sizes, distances, and shapes of the parameter-space partitions while conveying the partitions' uncertainties to avoid misinterpretations.

T4.4 Exploring the *simulation outcome*. Users should be able to explore the simulation outcome on different levels of detail to obtain a comprehensive understanding of the simulation ensemble. Thus, spatial and temporal visualizations need to be included in the analysis approach.

It is still common in practice to address these tasks by a manual approach where the ensemble members are analyzed and compared individually. As this process is tedious and time-consuming, we propose an interactive approach realized as an integrated tool that facilitates the entire workflow. Therefore, we incorporate (semi-)automatic algorithms for computing the parameter-space partitioning while allowing users to use their domain knowledge to influence the outcome. While different other approaches in visualization research address a subset of these tasks, none of them provides a comprehensive solution, as shown in Table 4.1. It becomes clear that our approach is the only one that fully supports all identified tasks.

4.3 OVERVIEW

We propose to use multiple coordinated linked views to address the tasks above and investigate the many different facets of the data. Each view addresses a subset of the tasks and facets, but the close integration allows for a comprehensive analysis. The analytical workflow follows the principle "Overview first, zoom and filter, then details on demand" [313] and includes different facets of the ensemble data as shown in Figure 4.1.

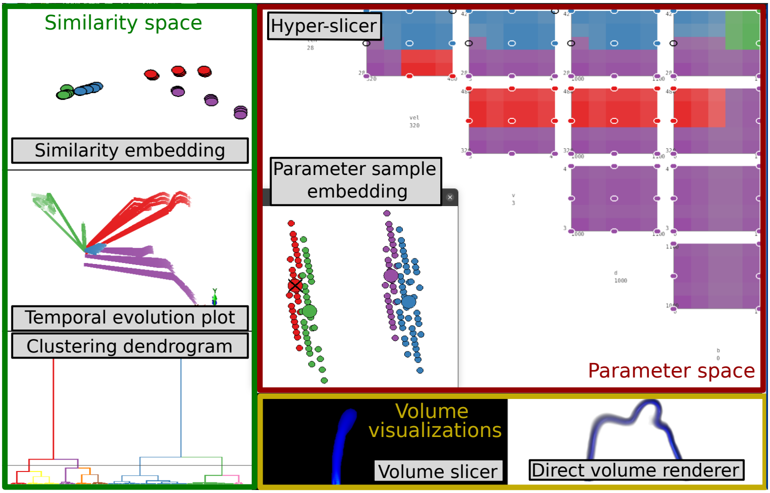


Figure 4.1: Our visual analysis approach allows for investigating the parameter space of a simulation ensemble together with its similarity space and direct volume visualizations.

At first, a clustering of the ensemble members based on their similarity needs to be created. Thus, we obtain a parameter-space partitioning (see task T4.1). Clusters can be computed fully automatically, as is commonly done in pattern recognition. However, any clustering algorithm depends on parameters significantly influencing the result. Completely manual cluster generation as performed, for example, by Bergner et al. [34] requires inspection of all individual ensemble members and is very time-consuming. Therefore, we opt for semi-automatic cluster generation where a clustering is computed automatically, but the driving parameter of the algorithm can be adjusted interactively. Hierarchical clustering approaches only depend on a pruning level determining how many clusters should be formed. This parameter can be chosen intuitively by the user, especially if the clustering hierarchy is visualized using a dendrogram. Thus, we choose a hierarchical clustering where the users can select and change the clustering level in a *clustering dendrogram* as described in more detail in Section 4.4.2. Alternative methods for finding clusters in the similarity space of the simulation ensemble depend on less intuitive parameters such as kernel sizes, the (static) number of expected clusters, or bin sizes.

To judge and, if necessary, modify the clustering outcome, we include visualizations of the similarity space. We use a 2D embedding of the simulation runs based on their pairwise similarities. Thus, each point in the embedding represents one simulation run. As we want to investigate the similarities between the runs, they should be encoded as distances in the embedding. Therefore, we aim at preserving distances by minimizing the stress, which is done by multi-dimensional scaling [373]. In the resulting MDS embedding, which we will refer to as *similarity embedding*, the points are color-coded according to the cluster to which the corresponding simulation runs belong. For more details, see Section 4.4.3.

To address task T4.2 and obtain an overview of the entire parameter space and its partitioning, we include a segmentation embedding as discussed in Chapter 3 where the different segments are color-coded according to the clusters they belong to. In the context of this chapter, we refer to the corresponding visualization as *parameter-space embedding*, see Section 4.5.2 for details. The embedding of the parameter space preserves joint boundaries and allows for easily assessing if the segments in parameter space that are induced by the clustering are connected. The segmentation embedding further contains additional information, like the sizes of the segments. Besides providing an overview, this embedding also supports navigation. In the paper corresponding to this chapter [100], we proposed to use an MDS embedding of the parameter space samples. We provide a more detailed discussion of the two approaches in Section 4.5.2.

The overview visualization of the parameter space is complemented by a distortion-free visualization to analyze the geometric structure of single segments (see task T4.3). The embedding neither preserves the shapes of the segments nor allows for reading of the values in the different dimensions. Alternatives would be parallel coordinates or SPLOMs, but they hinder intuitive understanding by overplotting. A detailed comparison is provided in Section 4.6. Instead, we propose an extended version of the hyper-slices, which we refer to as *hyper-slicer* and is presented in Section 4.5.1. It provides distortion-free visualizations [266, 342] and is based on viewing slices, which is a standard visual encoding and familiar to many domain experts. To overcome the shortcoming that hyper-slices only show information around a single focus point, we enrich our visualization with projections of the boundaries of the parameter-space partitions and an uncertainty encoding of the partitioning.

Finally, one also wants to investigate the behavior of the ensemble members (see task T4.4). We use a time-dependent embedding of the similarity space to encode the evolution over time. Here, we use a 2D MDS embedding for the same reasons as for the similarity embedding discussed above. Each ensemble member is shown as a polyline that represents the temporal evolution. Therefore, we refer to this visualization

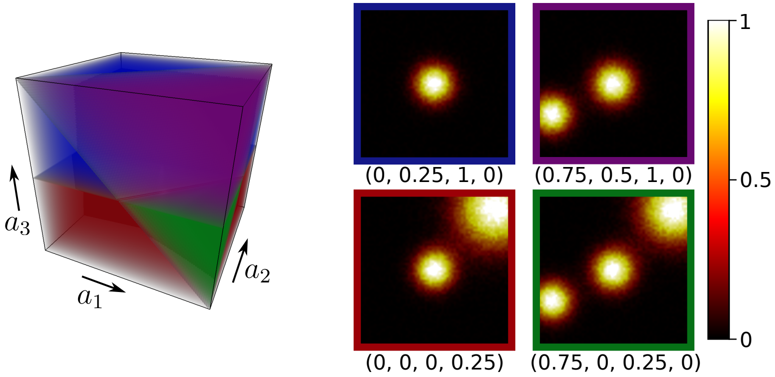


Figure 4.2: The synthetic dataset is created by partitioning the parameter space in four regions (red, green, blue, and purple) depending on the parameters a_1, a_2, and a_3 while the parameter a_4 does not influence the result. The images on the right show examples of the 2D data used as simulation output where the colored frames indicate to which cluster the ensemble members belong, and the exact parameter settings are given below the images.

as a *temporal evolution plot*, see Section 4.4.4 for more details. Single time steps of a single ensemble member can be shown by *direct volume rendering* or, in case of occlusion or for 2D data, using a *slice viewer*. For complex 3D data, it is also possible to combine those two visualizations.

To visually link the different views, we choose consistent color coding, because color is most intuitive for this purpose. For more details on linking the coordinated view, see Section 4.7. For the implementation of our approach, we use the Voreen framework [231, 85], which provides a modular structure. Thus, the different components can be reused for other applications, and the workflow is easily adaptable, for example, if it should be extended to vector field data.

4.3.1 *Synthetic Dataset*

To explain our visual encoding in the following parts and validate our approach, we use a synthetic dataset with a four-dimensional parameter space defined over $[0, 1]^4$. The four parameters are called a_1, a_2, a_3, and a_4, where only the first three influence the output. The simulation outcome is defined over a 2D domain of a resolution of 64×64 and is characterized by a varying number of Gaussian kernels that also differ in position and size. Here, we model four different behaviors as represented in Figure 4.2. The four different regions in parameter space are defined as follows:

1. Only one Gaussian in the center (blue)
2. One Gaussian in the center and one in the lower left (purple)
3. One Gaussian in the center and one in the top right (red)
4. One Gaussian in the center, one in the lower left, and one in the top right (green)

The 4D parameter space is sampled equidistantly with 5 samples in each direction leading to 625 ensemble members in total. The detailed equations for creating this dataset are presented in Appendix B.1.

4.4 SIMILARITY SPACE ANALYSIS

The analysis of the similarity space allows for a comprehensive analysis of the whole ensemble and a semi-automatic clustering of the ensemble members. The clustering is then used to induce a partitioning in the parameter space. Therefore, we will start by explaining our approaches for similarity space analysis before presenting the parameter-space visualizations that are the main contribution of this chapter.

4.4.1 *Clustering*

One main goal of the similarity space analysis is defining a clustering that induces a parameter-space partitioning in different regions that show similar behavior in the simulation outcome. Therefore, we will first present our choice for a similarity measure (see Section 4.4.1.1) before discussing the choice of the clustering algorithm (see Section 4.4.1.2).

4.4.1.1 *Similarity Measure*

The first part of the analysis is based on the similarity space. To define the similarity space, one needs to define a pairwise similarity measure between ensemble members. The field similarity measure proposed by Fofonov and Linsen [111] defines the similarity between two scalar fields and can be seen as a generalization of an isosurface similarity measure to include all possible isosurfaces. They discuss that their similarity measure is preferable for ensemble data as it preserves characteristics of observed phenomena. We refer to their paper for a detailed comparison to other similarity measures.

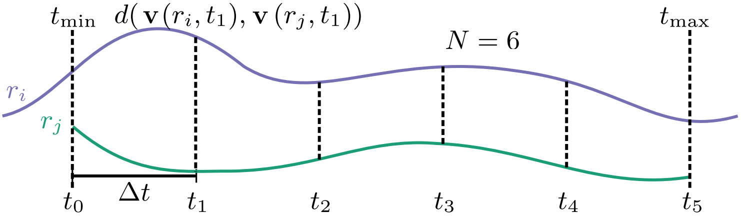


Figure 4.3: The illustration shows how the similarity between two ensemble members r_i and r_j can be computed over time. Only the distances in the joint time interval are considered.

In the first step of computing the similarity between two timesteps t_m and t_n of runs r_i and r_j, they create a vector of random sample points using a Monte Carlo approach. The Monte Carlo sampling significantly reduces the computation times compared to using all sample points in the original data. From these sample points that are equal for all considered fields, a vector $\boldsymbol{v}(r_i, t_m)$ that represents the scalar field of run r_i and timestep t_m can be created (see also Section 2.3.2). Now, the distance $d(\boldsymbol{v}(r_i, t_m), \boldsymbol{v}(r_j, t_n))$ between the two vectors $\boldsymbol{v}(r_i, t_m)$ and $\boldsymbol{v}(r_j, t_n)$ can be computed as

$$d(\boldsymbol{v}(r_i, t_m), \boldsymbol{v}(r_j, t_n)) = 1 - \frac{\sum_k (1 - \max(v_k(r_i, t_m), v_k(r_j, t_n)))}{\sum_k (1 - \min(v_k(r_i, t_m), v_k(r_j, t_n)))}, \quad (4.1)$$

and corresponds to the distance between the scalar field of run r_i at timestep t_m and run r_j at timestep t_n.

For clustering simulation runs, we need to generalize this similarity measure to cover temporal data. For fast computation, we use the distances created between the different time steps following Equation 4.1. Therefore, we consider the average distance over time as shown in Figure 4.3. We need to consider that different simulation runs might cover different time intervals and use different step lengths. Therefore, we only consider the overlapping time interval $[t_{\min}, t_{\max}]$. As we consider simulation ensembles here, we can assume that the overlapping time interval is sufficiently large. Additionally, pairwise comparisons between ensemble members allow for always choosing each pair's largest joint time interval. Next, the joint time interval is equidistantly resampled. The number of sample points N is chosen as the maximum number of timesteps of the two runs that should be compared. It is used to determine the step length

$$\Delta t = (t_{\max} - t_{\min})/(N - 1).$$

Thus, we can compute the distance between runs r_i and r_j as

$$d(r_i, r_j) = \frac{1}{N} \sum_{n=0}^{N-1} d(\boldsymbol{v}(r_i, t_n), \boldsymbol{v}(r_j, t_n)), \tag{4.2}$$

where $t_n = t_{min} + n\Delta t$. Computing the distance based on the pairwise distance between the individual timesteps allows for effective computations if the distances between all time samples are precomputed. Therefore, we can also support selections of temporal intervals of interest for which the pairwise similarities over time can be recomputed at interactive rates.

However, simulation runs are not always temporally aligned. Therefore, depending on the application scenario, it should be possible to consider a time lag between the ensemble members to find the best match. That is why we allow a time shift to a user-defined upper threshold τ_{max}. If the time shift should be included, the distance between the runs r_i and r_j is computed as

$$d(r_i, r_j) = \frac{1}{N} \min_{\tau} \left(\sum_{n=0}^{N} d(V(r_i, t_n), V(r_j, t_n + \tau)) \right), \tag{4.3}$$

instead of using Equation 4.2. Here, τ is varied in discrete steps in the interval $[-\tau_{max}, \tau_{max}]$. The choice of using either Equation 4.2 or Equation 4.3 is left to the domain expert as it is usually clear from the application scenario.

Based on the pairwise distances computations, one can form a distance matrix D_R, which stores the distances among all runs and forms the similarity space.

4.4.1.2 *Hierarchical Clustering*

To find clusters of similar ensemble members (see task T4.1), we use the distance matrix D_R as an input for clustering. Here, we propose to use hierarchical clustering because the number of clusters is usually initially unknown. Hierarchical clusterings generate hierarchies for different numbers of clusters that could then be changed interactively. For this purpose, clusters are iteratively merged following a bottom-up approach until one obtains a single cluster containing all data points.

The structure of the hierarchical clustering can be represented as a tree data structure where each node is formed by a cluster created by merging the clusters represented as its children. Thus, the single cluster containing all sample points forms the root of the tree, while the leaves are formed by clusters that contain only single simulation runs. By selecting a pruning level corresponding to a certain height of the tree, the number of clusters can be selected after computing the complete hierarchical clustering.

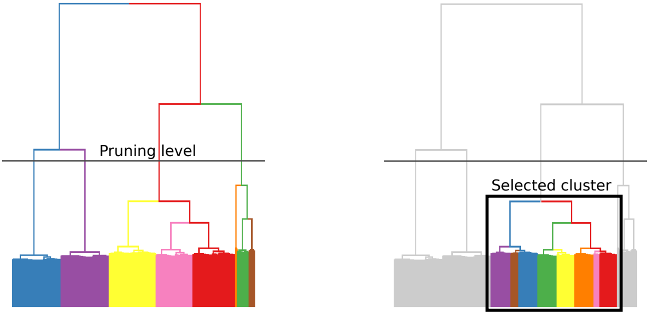
The clusters that should merge when creating the hierarchical clustering are determined by the linkage algorithm. However, the choice of the best linkage algorithm depends on the data. Thus, we included a set of common linkage algorithms and left the choice to the user. It also allows for interactively experimenting with different algorithms. In our approach, we include the following algorithms:

- Ward's minimum variance method (ward.D2) minimizing the increase in variance when merging two clusters
- An adapted version of Ward's minimum variance method (ward.D) where the distances are not squared
- Single linkage, which uses the minimum distance
- Complete linkage, which minimizes the maximum distance between points
- Average linkage with the unweighted pair-group method for arithmetic averages (UPGMA) minimizing the average distance between pairs of elements when considering the union of the clusters
- Average linkage with the weighted pair-group method for arithmetic averages (WPGMA) based on the average distance between pairs of points from both clusters

We used `R` to compute the linkage algorithm, where we used `RInside` [88] to access it from our C++ implementation.

4.4.2 *Cluster Analysis*

The result of the hierarchical clustering is not a unique clustering but instead a tree, as described in the previous section. A concrete clustering can then be obtained by defining a height in the tree and using it as a pruning level. We follow the standard procedure of visualizing the clustering result as a dendrogram to find a suitable pruning level and obtain an overview of the hierarchical clustering. An example can be seen in Figure 4.4a. The dendrogram directly shows the tree structure and, thus, visualizes the order in which the clusters merge. Additionally, the height in the tree encodes the dissimilarities between the merging clusters based on the linkage algorithm. When selecting a pruning level,



(a) Full dendrogram. (b) Cluster selection.

Figure 4.4: The pruning level in the clustering dendrogram can be used to define the number of clusters and change it interactively (a). Selecting single clusters in the dendrogram, induces a re-coloring of the corresponding clusters and allows for more detailed investigation (b).

the number of clusters is set based on the number of branches cut at this height. For the example shown in Figure 4.4a, the chosen pruning level results in 4 clusters.

When choosing a pruning level, the resulting clustering is computed and linked to the other visualizations by assigning colors to the clusters. For defining the colors for each cluster, we define three requirements:

1. The colors should be easily distinguishable.
2. Very high numbers of clusters should be avoided to prevent an oversegmentation of the ensemble and, later on, the parameter space. Therefore, we can assume the need for a moderate number of distinguishable colors.
3. When changing the granularity of the clustering by adapting the pruning level, the colors should only change as much as necessary.

Following these requirements, we choose a categorical color map. Here, we decided to use a color map with up to 12 different colors generated by ColorBrewer [51]. For the results in this chapter, we used the color map "Set 1" with 8 different colors. Note that the color maps could be easily adapted or exchanged if a different number of colors is needed or if, for example, color blindness should be considered.

We traverse the tree following a top-down approach to assign the colors to the clusters. The first color gets assigned to the root node. For each node, the same color is propagated to the child with the largest subtree

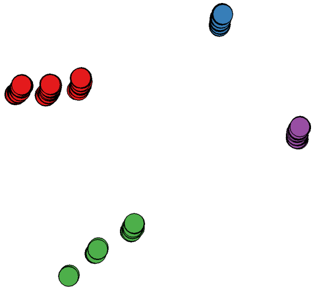


Figure 4.5: Each point in the similarity embedding represents one ensemble member, and the distance between the points encodes the similarity between the ensemble members. The color encodes the clusters based on color assignment in the dendrogram.

such that the cluster with the most samples keeps the color if the pruning level is changed. The other child obtains the next color in the color map until no more colors are available. In this case, both children obtain the parent's color.

This coloring procedure leads to many clusters getting the same color if small clusters should be investigated. This prevents an oversegmentation of the parameter space. However, if clusters on a lower level of detail should be investigated, subclusters can be selected. To the resulting sub-tree, the color is reassigned following the same method, while the clusters that are not selected are colored in light gray to provide orientation while not being too prominent. An example of a dendrogram with this structure is shown in Figure 4.4b where the recoloring compared to Figure 4.4a is shown.

4.4.3 *Similarity Embedding*

Besides visualizing the structure of the clustering in the dendrogram, we also want to visualize the similarity space as given by the distance matrix D_R. It allows for evaluating the clustering quality directly in the space where the clustering is created. Here, we use an embedding where the distances in the low-dimensional space should reflect those in the multidimensional space defined by D_R. This goal is achieved by minimizing the stress function, which is the objective function of MDS, which we apply to the distance matrix D_R. Thus, each projected sample point in the embedding represents one ensemble member.

To visually link the embedding to the other visualizations and show the clusters, we color-code the sample points according to the assigned

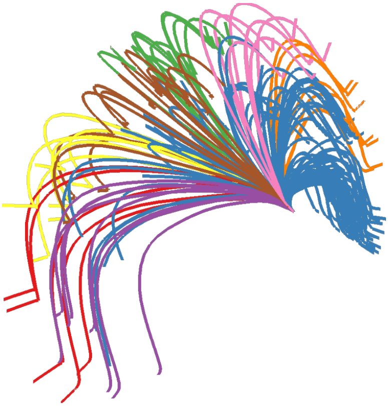


Figure 4.6: In the temporal evolution plot, each run is represented as a curve over time. Distances between points on the curves represent similarities in the simulation outcome.

cluster colors as shown in Figure 4.5. If the clustering is changed in the dendrogram, the new clustering is immediately reflected in the embedding by changing the colors. Thus, a suitable clustering can be found by interacting with the dendrogram and investigating how well the clustering matches potential clusters in the embedding.

In the synthetic dataset, whose embedding is shown in Figure 4.5, we observe four groups in the embedding as expected by the definition of the dataset. Therefore, we also select a pruning level that leads to four clusters, as shown in Figure 4.4a. When varying the pruning level, one could also split the green and red clusters into three subgroups that align with the subgroups visible in the embedding.

4.4.4 *Temporal Evolution Plot*

The similarity embedding is based on the distance matrix D_R, which is aggregated over time. However, to investigate the runs' behavior over time, it is essential to visualize their temporal evolution. For example, this also allows for finding out whether runs converge or diverge and observing behavior like periodicities (see task T4.4).

For visualizing the temporal evolution, we use multi-run plots [112] as described in more detail in Section 2.3.2. For creating the multi-run plot, we use the distance matrix D_T that contains the distances between the individual timesteps and is used as a starting point to aggregate for the distance matrix D_R.

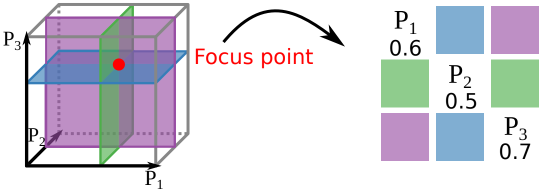


Figure 4.7: The slices of the hyper-slicer are created around a focus point (red). The parameter space is formed by parameters P_1, P_2 and P_3 (shown as gray cube). The slices are then created from the volume formed by the parameter space. The coordinates of the focus point are shown together with the parameter names on the diagonal of the matrix.

Similar to the similarity embedding, the different runs in the temporal evolution plot are color-coded according to the cluster they belong to. This allows for quickly evaluating how the runs of certain clusters behave over time, as shown in Figure 4.6, which shows an embedding in 3D. In general, we support embeddings in 1D, 2D or 3D spaces where the optimal dimensionality depends on the analysis goal as well as on the intrinsic dimensionality of the data. When creating a 1D embedding, the embedding can be shown on the vertical axis while time is used as the horizontal axis. Thus, the evolution over time can be intuitively analyzed. Explicitly encoding the time as an axis also allows for selecting time intervals for further analysis. This is especially helpful if, for example, initial transition phases should be excluded. In 2D and 3D embeddings, the time component can be included by varying the saturation of the color. Alternatively, it is possible to encode time intervals that were selected in 1D by rendering the timesteps inside of the interval in full saturation and the remaining time steps with decreased saturation.

4.5 PARAMETER-SPACE VISUALIZATIONS

The main contribution of this chapter lies in visualizing the parameter-space partitioning that is induced by the clustering in the similarity space as described in Section 4.4.1.

4.5.1 *Hyper-slicer*

For addressing task T4.3 and showing an undistorted view of the parameter space that also includes to directly read of the parameter values, we extend the hyper-slices as proposed by van Wijk and van Liere [353]. In the following, we will refer to our enriched hyper-slice visualization as *hyper-slicer*. Hyper-slices aim at visualizing a multi-dimensional space by showing an axis-aligned slice for pairwise combination of dimensions. Similar to SPLOMs, the slices are shown in a matrix layout but they do not show a projection of the data but instead only show the data that is in the slice. Thus, they can also be seen as a multi-dimensional generalization of slice viewers equivalent to SPLOMs being a multi-dimensional variant of scatterplots.

To apply hyper-slices to a multi-dimensional parameter space, we see each parameter as forming an axis of this space. Then, the parameter space can be sliced around a focus point that corresponds to one set of parameter values and can be changed interactively by the user. Creating a hyper-slice visualization for 3D data around a focus point is shown schematically in Figure 4.7. The resulting hyper-slices show a 2D plane for each pairwise combination of axes where each of the planes contains the focus point. In our visualization, we directly encode the values of the focus point for the parameters on the axes of the slice by a gray cross. Thus, we provide some orientation in the multi-dimensional space. To obtain the exact information about the focus point in all dimensions, we show the corresponding parameter values on the diagonal below the names of the parameters. The hyper-slicer provides zooming and enlarging single slices, allowing for a more detailed investigation of the slices.

In general, hyper-slices, like SPLOMs, do not scale well when the number of dimensions increases. Therefore, we include an automatic reordering of the dimensions based on the correlation between a single parameter and the simulation result. Thus, users can focus on the most relevant dimensions. To facilitate this, manually hiding a set of dimensions is also possible, leading to a reduced dimensionality in the visualization.

4.5.1.1 *Partitioning Visualizations*

As we want to use hyper-slices to show the partitionings, we adapt the approach. To show the partitioning information, we visualize to which segment the samples belong, the boundaries of the segments, and their uncertainties. Our different visual encodings are illustrated in Figure 4.8.

The parameter-space samples are drawn as dots. We differentiate between sample points that belong to the currently shown slice by showing them with a white outline, while other samples are shown with a black outline. Showing all sample points, including those outside the slices, provides context about their location in the parameter space. For samples located on the slice, we color-code them according to which cluster

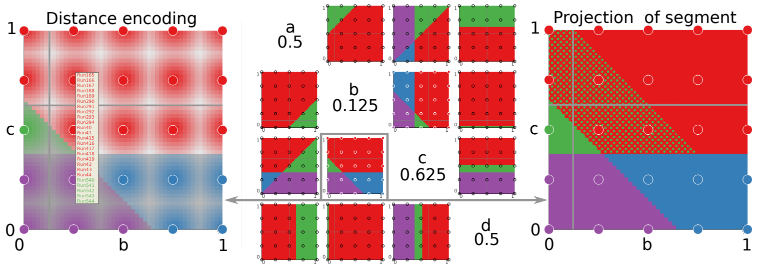


Figure 4.8: The hyper-slicer (center) shows the parameter space of the synthetic dataset. The sample points with white circles lie inside the slice, while those shown as black circles are located outside of the slice. The focus point is shown as gray lines, and the colors indicate the different segments. The slices can be enriched with an uncertainty visualization by changing the saturation (left) or a projection of the segment boundaries of selected clusters shown using a dot texture (right).

they belong, where we use the colors as assigned in the dendrogram and which are also used in the other visualizations.

However, in the case of parameter spaces that are sampled on an axis-aligned grid, multiple samples might overlap because they are projected to the same positions in the slices, as shown in Figure 4.8. To show which sample points are projected to the corresponding position, the users can hover over the points, and a list of all points in that position is shown. In that list, we color code the names of the ensemble members by using the cluster colors. Overall, the encoding of the sample points resembles a SPLOM in which the points lying in the slices defined by the hyper-slicer's focus point are highlighted.

To visualize the parameter-space partitioning, we color-code the corresponding slices and use the cluster colors as defined in the dendrogram. The clustering of the similarity space can be transferred to the parameter space by assigning the parameter-space samples to the same clusters as their simulation outcome. To obtain a dense partitioning of the parameter space, we use multi-class support vector machines (SVMs) where we use the implementation provided by libsvm [59]. The SVM is trained based on the clusters assigned to the sampling points. For the kernel, we use radial basis functions while we leave the choice of the parameters C (cost for wrong classifications) and γ (can be interpreted as the range of influence of single samples) to the users. As wrong classifications are undesirable, we show a warning in case they occur. Then, the users can adapt the parameters to find a more reasonable partitioning.

An alternative to computing the partitioning in parameter space would be the approximation of a Voronoi diagram. However, using SVMs leads to smoother boundaries, especially in the case of sparsely sampled parameter spaces.

We discretize a regular grid to obtain a dense partitioning of the parameter space. Then, we use the SVM to determine the cluster to which the grid points belong. The resolution of the grid can be chosen interactively by the users depending on the analysis goal. As parameter-space samplings are often sparse due to the high simulation costs, a relatively low grid resolution is often sufficient. While lower resolutions are computationally less expensive, they might induce stair-case artifacts as shown in Figure 4.8. While higher grid resolutions reduce the artifacts, they are also computationally more expensive and might lead to short delays. After assigning each grid point to a cluster, the partitioning in the selected slice can be shown directly analogous to a slice viewer.

Independent of the method, the computed partitioning and its boundaries are only an approximation. As the simulation results between the existing sample points are unknown, it is unavoidable to compute further simulation for more exact boundaries.

4.5.1.2 *Uncertainty Visualization*

To provide information about the uncertainty in the corresponding regions of the parameter spaces, it should be directly included in the visualization. This also provides hints to the domain experts on which parameter-space regions are undersampled and benefit most from additional simulations. The uncertainty in the partitioning is mainly caused by assigning the color of the cluster to each grid point independent of the distance to the nearest parameter-space sample. Thus, some parts of the parameter space are assigned to partitions despite having no samples in this region. Therefore, we aim to visually encode these regions by adding an uncertainty visualization where we encode the Euclidean distance to the nearest sample point.

To encode the distances, we vary the saturation of the colors used to show the partitioning. As the different parameter values might span very different ranges, we normalize them before calculating the distances. After computing distances for all grid points, we also normalize the distances to the interval $[0, 1]$. Then, we can modify the saturation for each sample by multiplying the color values by $1 - d$ where d is the normalized distance value. Thus, the most distant grid points get assigned desaturated colors, while the grid points in the same position as parameter sample points get assigned a fully saturated color. An example of a slice that shows the variation in saturation based on the distances is presented in Figure 4.8 (left). In this regularly sampled parameter space, one can observe that the uncertainty is highest between the sample points leading to the least separation. Close to the sample points, the colors show the highest saturation.

While the uncertainty encoding of the regularly sampled parameter space meets our expectations, it is beneficial for irregularly sampled parameter spaces where the position of the nearest sample point is unclear. Besides showing uncertain regions, the segment boundaries become less clear if they are very distant from any parameter-space sample. This prevents the users from overinterpreting them even though they are uncertain. However, as this might introduce distractions, the feature can also be turned off if the uncertainty is not important for the current analysis task.

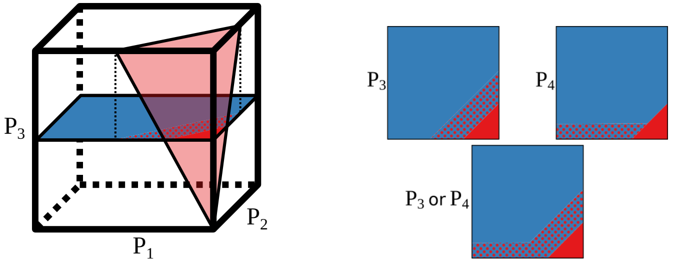
4.5.1.3 *Boundary Projection*

As the hyper-slice visualization only shows the partition around the focus point, it only shows a small fraction of the parameter space. For investigating the whole parameter space and understanding the extent of specific segments, a large amount of time-consuming exploration would be necessary. Additionally, many manual changes of the focus point impose a high cognitive load. Therefore, our hyper-slicer includes the possibility of projecting the boundaries of a selected segment to the currently shown slice. This hints towards the parameters that need to be varied to investigate the boundaries of the parameter-space segments.

To encode the extent of the segments, we use a dotted texture that resembles stippling. We overlay a texture that contains dots on a grid that is rotated by 45°. The dots are encoded in the same color as the selected similarity cluster to provide a visual extension to the corresponding segment. At the same time, the segmentation shown in the slice remains visible.

For defining the region in which the texture overlay needs to be shown, we create a binary mask b_{ij}^k for the projection along parameter P_k in the slice spanned by P_i and P_j with $i, j, k \in \{1, ..., n\}$ for an n-dimensional parameter space. We assume $i < j < k$ without loss of generality and denote $(p_1, ...p_n)$ the current focus point where p_i is a parameter value in the range spanned by parameter P_i. The mask can be computed for each point (x, y) as

$$b_{ij}^k(x, y) = \begin{cases} 1 & \text{if } \exists z : l(p_1, \ldots, x, \ldots, y, \ldots, z, \ldots, p_n) = s_m \\ 0 & \text{otherwise} \end{cases}$$



(a) Boundary projection.
(b) Boolean operations.

Figure 4.9: The schematic drawing illustrates the projection of the boundaries of the red segment on the slice spanned by parameters P_1 and P_2. a) The red plane separates the red and the blue segment. The projection on the plane is shown as a dotted texture. b) Projections along different dimensions formed by different parameters yield different results. Projection along different parameters can be combined by Boolean operations, like here for the operation P_3 *or* P_4.

where $l(p_1, ..., p_n)$ is the parameter-space label at the point $(p_1, ..., p_n)$ and s_m the label of the segment. The mask indicates whether the grid cell around point (x, y) shall exhibit the texture.

The projection is illustrated in Figure 4.9 on the example of a 3D parameter space. In this example, the red plane partitions the parameter space into two segments. Figure 4.9a shows how the red segment is projected on the plane spanned by parameters P_1 and P_2 along the axis of P_3. The boundaries are projected on the slice, and the shape's interior is filled with a red-dotted texture.

The resulting slice is also shown in Figure 4.9b. Assuming that the dataset contains a fourth parameter P_4, the projection could also be performed along that direction, potentially resulting in the projection shown in the top right of Figure 4.9b. This methodology allows for investigating projections along single dimensions. However, it might also be desirable to investigate several parameters at once. Therefore, we allow for combining different projections using Boolean operations. For example, for considering the union of the projections along P_3 and P_4, the Boolean expression "P_3 *or* P_4" can be used, which results in a visualization as shown in Figure 4.9b in the bottom.

For evaluating a Boolean expression, the binary mask for each parameter is computed. The operations are then computed on the masks. For example, for the union discussed above, the *or*-operation is applied on the masks for parameters P_1 and P_2. Arbitrary expressions are possible where all parameters can be included. However, the dimensions shown on the axes are excluded for the respective slice. In our tool, the most common Boolean operators like *and*, *or*, *not*, *xor*, *nor*, *nand*, implication and equivalence are included. A Boolean expression using the parameters and the operators can be directly entered as text to the graphical user interface. To facilitate the projection of the entire parameter space, which is the most common application, we also introduce the keyword *Complete*, which can be entered as a Boolean expression. By using this keyword, the union over all parameters is shown.

Figure 4.8 (center) shows the hyper-slicer for the synthetic dataset with the uncertainty visualization and the boundary projection turned off. The slice spanned by parameters a_2 and a_3 includes all clusters. Therefore, we choose this slice to verify our enrichments and enlarge the slice for a more detailed analysis. This slice with an activated uncertainty visualization is shown on the left. As previously discussed, the saturation decreases when the distance to the parameter-space samples increases and reaches its minimum between the sample points, where the distance to all neighbors is the largest. When hovering about one of the sample points, the corresponding points are shown in an overlay. Based on the colors, we can see that in this position, points belonging to the red cluster and those belonging to the green cluster are projected. When selecting the green cluster and activating its projection, the slice shown in the right of Figure 4.8 is obtained. The cluster only occurs for smaller values of a_2 and larger values of a_3, and the boundary is diagonal, which fits the definition of the dataset that can be seen in Figure 4.2.

4.5.1.4 *Reducing the Dimensionality*

Similar to SPLOMs, hyper-slices do not scale well with the dimensionality of the data. For the application to parameter spaces of spatio-temporal simulation ensembles, the dimensionality usually does not become very large. However, limiting the number of dimensions to reduce the number of slices shown in the matrix might still be desirable. For this purpose, we allow for reducing the dimensionality of the parameter space by allowing the users to select the parameters that should be shown. To facilitate the selection, we support re-ordering the parameters based on their influence on the simulation outcome. The influence is measured by the absolute value of the Pearson correlation coefficient C_{P_i} between the parameter values P_i and the first principle component of the similarity embedding. Thus, the correlation coefficient can be computed as

$$C_{P_i} = \frac{\sum_{j=1}^{n}(P_i(r_j) - \mu_{P_i})(\nu(r_j) - \mu_\nu)}{\sqrt{\sum_{j=1}^{n}(P_i(r_j) - \mu_{P_i})^2}\sqrt{\sum_{j=1}^{n}(\nu(r_j) - \mu_\nu)^2}},$$

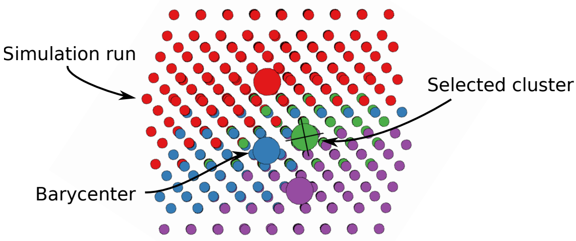


Figure 4.10: Each point in the parameter sample embedding encodes one simulation run. The distances encode distances in parameter space, and the color show to which cluster the corresponding simulation run belongs. The barycenters of the points are shown as larger points and can be used for navigating the hyper-slicer. Selected clusters for in-detail investigation in the hyper-slicer are marked with a cross.

where $\nu(r_j)$ is the first principle component of the similarity embedding for ensemble member r_j and μ_{P_i} and μ_ν are the means of $P_i(r_j)$ and $\nu(r_j)$, respectively. The correlation values can be visualized using a bar chart. This allows the users to investigate the correlation values and decide which and how many dimensions should be included in the analysis.

4.5.2 *Parameter-Space Embedding*

While the hyper-slices allow for detailed visualization of the segments, it does not provide a complete overview. Investigating the whole parameter space would require a large amount of interaction. Therefore, we include an embedding of the parameter space (see task T4.2) that provides an overview and facilitates navigation.

We experimented with two options for the parameter-space overview visualization. In the paper presenting the hyper-slicer [100], we use an MDS projection of the parameter-space samples to create a 2D embedding of the parameter space which we refer to as *parameter-sample embedding*. For computing the embedding, the distances in the parameter space are used. This embedding directly shows the parameter-space samples and allows for investigating the distribution of the parameter sample points belonging to the different clusters. Thus, the clustering quality can also be evaluated in parameter space, and one obtains a first impression of how the samples corresponding to a particular cluster are distributed in parameter space.

This approach is shown as an annotated example in Figure 4.10. Each point of the embedding defines a simulation run where the color represents the cluster color. An additional point, shown in increased size, represents the barycenter of the cluster. The primary purpose of these barycenter points is to facilitate the selection of the cluster's center as a focus point for the hyper-slicer. However, including the barycenters as additional information should be independent of the projection layout. Otherwise, changing the clustering would also modify the projection, which is undesirable. Therefore, the barycenters are computed in the projection instead of in the multi-dimensional parameter space.

For the synthetic dataset, the parameter-sample embedding shown in Figure 4.10 shows a regular structure induced by the parameter-space sampling. Despite the overlap induced by the dimensionality reduction, one can see that the segments seem to be connected, which meets our expectations. Even though the hyper-slicer allows for resolving the overlap, identifying connections between the different segments would be more difficult for more complex datasets.

Therefore, we propose using the segmentation embedding discussed in Chapter 3 as an alternative encoding. The regular grid used to compute the partitioning can be directly used as an input to the algorithm. However, as the clusters induced in similarity space are not necessarily connected in parameter space, each segment needs to get assigned a unique ID. After computing the embedding, all segments could be colored according to the cluster colors set in the dendrogram.

The *parameter-space partitioning embedding* directly encodes joint boundaries between the different segments and visually encodes the sizes of the segments and the lengths of the joint boundaries. It also allows for unambiguously identifying whether the clusters in similarity space transfer to connected regions in parameter space or to a set of separated segments. Further, the interaction of selecting the low-dimensional representation of a segment to switch to its barycenter would be more intuitive than selecting representative points that only vary in size compared to the sample points. On the other hand, the segmentation embedding does not show the parameter-space samples and their distances to each other, which is directly encoded in the MDS embedding of the parameter-space samples. Nevertheless, we propose to use the segmentation embedding as it provides a better overview of the parameter space and, thus, better fulfills task T4.2. However, as both approaches are included in the approach, the users could also interactively switch to the parameter-sample embedding.

The partitioning embedding of the 4D parameter space of the synthetic dataset is shown in Figure 4.11. One can see that the red segment

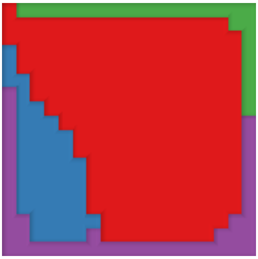
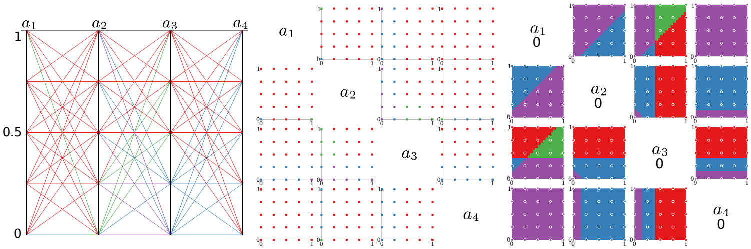


Figure 4.11: The parameter-space partitioning embedding of the synthetic dataset reveals its topological structure and the sizes of the respective segments.

is the largest, and the green segment is the smallest one. Additionally, all segments are connected to the boundary. Further, the four clusters in the similarity space also form clusters in the parameter space. These observations agree with the definition of the dataset. While the sizes of the segments can be estimated in the parameter sample embedding shown in Figure 4.10 based on the number of samples belonging to the corresponding clusters, the segmentation embedding takes the relative sizes of the partitions into account, which might deviate from the relative number of sample points, especially for irregular samplings. However, the very regular sampling structure, clearly visible in the parameter sample embedding, is not encoded in the segmentation embedding as it does not contain the parameter-space samples.

4.6 COMPARISON TO ALTERNATIVE VISUAL ENCODINGS

Our hyper-slicer visualization aims to visualize multi-dimensional data for which also other methods exist. Therefore, we will directly compare the hyper-slicer to SPLOMs and PCPs, which are the most common techniques for the (interactive) visual analysis of multi-dimensional datasets. Here, we focus on a comparison to the basic version of these visualizations even though several improvements in different directions exist. However, our approach could also be derived from SPLOMs instead of hyper-slices if one starts with the parameter-space samples and adds the visualization of the partitioning. Comparing against the base variants ensures that these variants are well known and used in practice, while more advanced and feature-rich variants might be rare, or we might even introduce new visual encoding against which we would compare. In addition, we only compare the visual encoding of the hyper-slicer, not our whole analysis system.

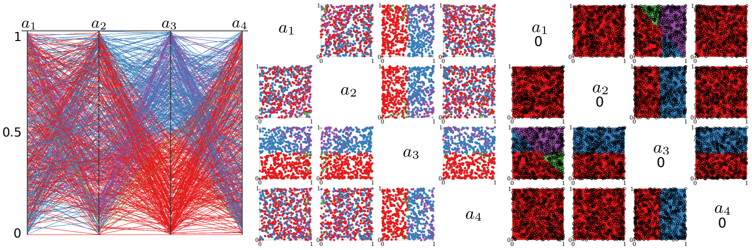


(a) PCP. (b) SPLOM. (c) Hyper-slicer.

Figure 4.12: The direct comparison of PCP (a), SPLOM (b), and hyper-slicer (c) for a regularly sampled parameter space reveals problems with overplotting in the first two approaches.

Parameter spaces are often sampled on structured grids like for the blood flow dataset (see Section 4.8.1) or the microswimmer dataset (see Section 4.8.3). Therefore, we start by comparing the visualization of regularly sampled parameter spaces on the example of the synthetic dataset as shown in Figure 4.12. In the PCP and SPLOM, the lines and points are color-coded according to the clusters, analogous to the hyper-slicer. Both PCP and SPLOM suffer from significant overplotting, which can be explained by regular sampling. Especially for PCP, many lines are drawn on top of each other because all possible combinations of parameter settings are present in the dataset. This also explains why no (visible) purple line exists between the axes representing parameters a_1 and a_2. Besides the problem that some lines and points are not visible in both visualizations, it is not clear how many points or lines are drawn on top of each other, and the rendering order strongly impacts the visual output. Therefore, it is impossible to decide if the red cluster only occurs for small values of a_3 or if this observation is due to overplotting.

In the hyper-slicer, however, the parameter ranges spanned by the segment could be derived by using the boundary projection. Additionally, the rendering order is very clear, as only a subset of the data is shown immediately due to the slicing of the parameter space. Only from observing the visualization, it is also not clear how many sampling points are projected on each position as the sampling point visualization resembles a SPLOM. Our approach allows hovering over the sampling points to deduce this information. This is an addition that could also be added to PCP and SPLOM, but solving the other issues in these visualizations is not straightforward. Order-independent blending or transparency might provide starting points to mitigate these issues, but these methods could also lead to mixed colors that are difficult to interpret.



(a) PCP. (b) SPLOM. (c) Hyper-slicer.

Figure 4.13: The comparison of PCP (a), SPLOM (b), and hyper-slicer (c) for a parameter space with unstructured sampling shows that PCP and SPLOM do not show the shapes of the segments as clearly as the hyper-slicer, see green and purple regions.

Additionally, the hyper-slicer facilitates obtaining a geometric understanding of the multi-dimensional space. It becomes more obvious when investigating parameter spaces with an unstructured sampling that occurred, for example, in the semiconductor dataset (see Section 4.8.2). The three visualization techniques are shown for irregular sampling in Figure 4.13. Even though overplotting is less prominent, it is still challenging to identify structures in the multi-dimensional space. Axis-aligned separations like the one between the red and the blue cluster caused by parameter a_3 can be identified easily in both visualizations. Nevertheless, already linear, diagonal structures for the boundaries between the purple and green clusters are barely recognizable. However, approaches such as edge bundling in PCP or density-based encodings in SPLOMs might reduce this problem but cannot fully overcome them. Zhou and Weiskopf [400] proposed an explicit extension to show correlations in PCPs, but their approach only deals with linear behavior. In real-world data, one cannot assume the boundaries to be linear. Therefore, we opt for showing only slices of the data around a focus point and, thus, focussing on a subset of the data which can be changed interactively. While the projection of the boundaries still provides the option to show data outside of the slices, interactively changing the focus point and slicing through the volume facilitates building a geometric understanding.

4.7 ANALYTICAL WORKFLOW

In this section, we discuss a typical analytical workflow on a more abstract level, while concrete application examples are presented in Section 4.8. A typical analysis, as depicted schematically in Figure 4.14,

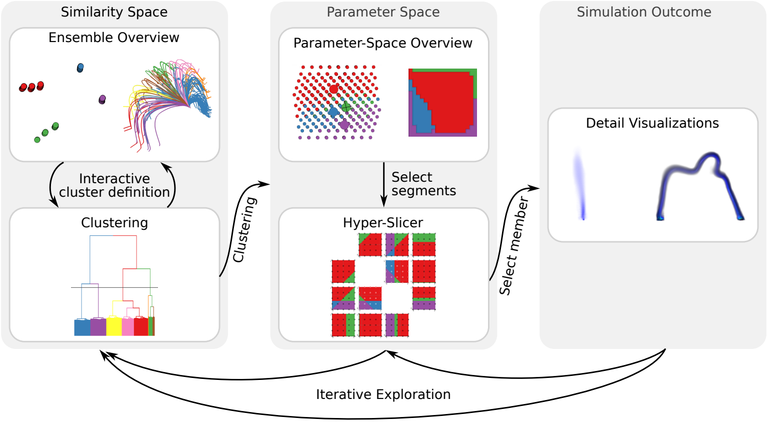


Figure 4.14: The analytical workflow allows for interactive visual analysis of different levels of details.

starts with obtaining an overview of the simulation ensemble for which the similarity embedding and the temporal evolution plot can be used. When investigating the temporal variation over time, the users might identify time ranges of particular interest and select them in the temporal evolution plot. The clustering dendrogram can be used to find a suitable clustering which is changed by varying the pruning level and the linkage algorithm when investigating the resulting clusters in the similarity embedding until a suitable clustering is identified.

The induced parameter-space partitioning can be checked in the partitioning embedding, where the users can identify if the parameter-space partitions are connected and study the topology of the partitioning together with the segment sizes. Additionally, the parameter sample embedding can be used for observing which parameter sample belongs to which cluster. Both visualizations that provide an overview of the parameter space allow for identifying single segments that should be investigated in more detail in the hyper-slicer. Here, the segments' boundaries and extent are investigated in more detail. To reduce the dimensionality of the hyper-slicer, the correlation between parameter values and simulation outcome is used to re-order the hyper-slicer and deselect less interesting parameters. To easily navigate the hyper-slicer, clusters can be selected in the parameter sample embedding or segment embedding. The boundaries of a selected cluster can be projected on the slices to investigate the cluster's extent.

Ensemble members can be selected by clicking on the samples shown in the hyper-slicer to investigate the simulation outcome and understand the characteristic behavior of the clusters. The corresponding simulation run is then shown in a slice viewer or a volume renderer. These simulations can also be used to investigate the variation of single ensemble members over time by varying the visualized time step. In the interactive analysis, it is common to frequently switch between the different levels of detail and go from a higher level of detail back to a less detailed visualization, for example, to vary the clustering or adapt the chosen time interval.

4.8 CASE STUDIES

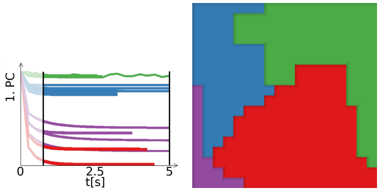
In the following, we will present three case studies from different domains to explain the practical applicability of our approach. All reported timings were obtained on a laptop with a 1.6GHz Intel Core i5 processor.

We also collected feedback in an informal setting from two professors and two graduate students involved in creating the three datasets. The feedback was collected in a single session which we started by providing a short introduction to our tool and the respective visualizations. After that, the domain experts worked with our tool to explore their datasets. Based on that, they also gave us feedback.

4.8.1 *Blood Flow*

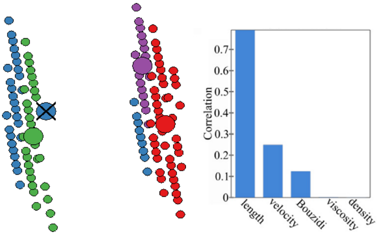
We first study a dataset that investigates the blood flow through vessels. Such numerical simulations can be used to define biomarkers that allow for identifying certain diseases [133]. The data analyzed in this work was created to simulate the flow through a brain aneurysm model to compare simulation models with measured data [204]. Finding the input parameters that best describe the experimental data is a key task in analyzing this kind of data. The simulation data analyzed in this work was created using a Lattice Boltzmann method. The dataset consists of 129 runs with 8 to 42 timesteps. We study the magnitude of the flow field on a regular grid with a spatial resolution of $128 \times 128 \times 128$. The simulation requires five input parameters: the viscosity of the fluid (*viscosity*), its density (*density*), a characteristic velocity (*velocity*), a characteristic length (*length*), and a parameter defining the boundary conditions (*Bouzidi*). The last parameter equals 1 if Bouzidi boundary conditions were used and 0 otherwise.

Thus, for this dataset, we analyze a 5-dimensional parameter space. The dataset was analyzed together with two domain experts. One of



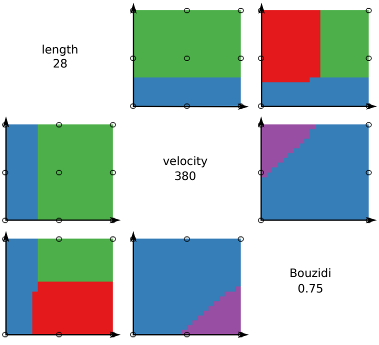
(a) Temporal evolution plot.

(b) Parameter segment embedding.



(c) Parameter sample embedding

(d) Correlation



(e) Hyper-slicer.

Figure 4.15: For the analysis of the blood flow dataset, the time interval shown in the temporal evolution plot (a) is selected. The parameter segment embedding (b) reveals that the four clusters correspond to connected regions in parameter space. However, the parameter sample embedding (c) shows that the parameter *Bouzidi* separates the parameter space into two groups, and the blue segment is the only one that occurs in both of them. Based on the visualization of the correlations between the first principal component and the parameters (d), the hyper-slicer (e) is limited to the three relevant parameters.

them created the simulation ensemble. The other one performed experiments that the simulation should describe.

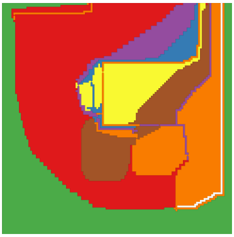
To compute the simulation runs' similarities, $32,768$ Monte Carlo seed points were used per timestep. When investigating the temporal evolution plot shown in Figure 4.15a, a transition phase in the beginning can be observed. Therefore, the following analysis is based on the simulation output starting at $0.7\,s$. Based on the distance matrix D_T for the different timesteps, the distance matrix D_R, which stores the distances between the runs, can be computed in $1.0\,s$.

After computing the similarities, the similarity plots and the dendrogram are used to find a suitable clustering. By interactively testing different linkage algorithms, Ward's minimum variance method (ward.D) was chosen as the most suitable as the identified clusters agree well with the clusters visible in the similarity embedding. The height shown in the clustering dendrogram (see Figure 4.1, lower left) directly points towards a prominent separation in two clusters which can be selected by adapting the pruning level. The resulting partitioning is induced by the parameter *length*, as can be found using the hyper-slicer. For further analysis, the partitioning is refined to contain four clusters.

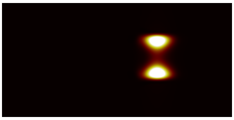
Next, the induced partitioning of the parameter space is investigated in the overview visualizations. In the parameter segment embedding shown in Figure 4.15b, one can identify that the violet segment is significantly smaller. Additionally, all segments are connected except the green and the violet ones. The samples are divided into two groups in the parameter sample embedding (see Figure 4.15c). When interacting with the hyper-slicer shown in Figure 4.15e, one can see that the division is along the parameter *Bouzidi*, which only takes the values 0 and 1. Only the blue segments occur in both parts and, therefore, for both parameter settings. Thus, the blue cluster is selected for further analysis in the hyper-slicer. When observing the correlation values between the first principle component of the output and the parameter values (see Figure 4.15d), only three of the parameter values lead to significant correlations. Therefore, the parameters *viscosity* and *density* are hidden in the hyper-slicer to reduce its dimensionality. The absolute values of the correlation also confirm that the parameter *length* is more influential than the parameter *Bouzidi*, which is surprising for the domain expert even though he expected the parameter *length* to be influential. The blue and violet clusters only occur for small lengths, which results in a high spatial resolution of the simulation domain. The boundary conditions do not separate those two clusters. As confirmed by a projection of the boundaries, the violet cluster only occurs for high velocities and without Bouzidi boundary conditions. This can be interpreted as a growing influence of the boundary conditions with an increased spatial resolution. For understanding the individual clusters, single ensemble members can be investigated in more detail using a direct volume rendering or a slice viewer, as shown in Figure 4.1 for this dataset.

4.8.2 *Semiconductor Quantum Wire*

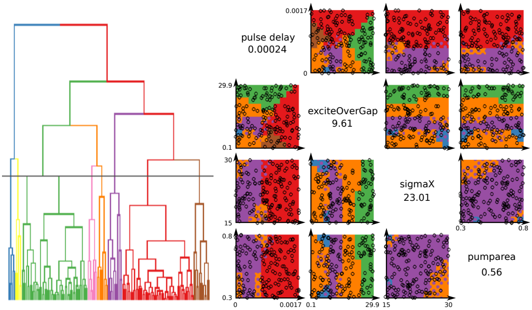
A quantum wire describes a wire where quantum effects influence the transport properties. When modeling these kinds of systems, they can be described as a one-dimensional spatial structure. The dataset used in this section investigates how light is emitted from a semiconductor quantum wire [358]. Here, it is interesting to observe the phase space spanned by the spatial dimension x and the additional dimension k leading to a two-dimensional domain.



(a) Segmentation embedding.



(b) Slice Viewer.



(c) Clustering dendrogram.

(d) Hyper-slicer.

Figure 4.16: The semiconductor dataset's parameter space is divided into 7 clusters based on the pruning level of the dendrogram (c). The parameter-space segments are shown in the embedding (a) and can be investigated in more detail in the hyper-slicer (d), where the purple cluster is selected for projection. It is characterized by two separate peaks with a small connection (b).

The simulation model depends on four different parameters in total, namely the energy difference between the laser pulse and the bandgap (*exciteOverGap*), the spatial variance of the laser pulse (*sigmaX*), the delay between the laser pulses (*pulseDelay*) and the pulse area (*pumparea*). The simulation ensemble consists of 150 runs with 70 timesteps each, where the parameter space was sampled randomly. The resulting two-dimensional scalar field for the phase space has a spatial resolution of 300×200. An exemplary heatmap visualization of the phase space is shown in Figure 4.16b.

We used $16,384$ Monte Carlo seed points per timestep to compute the similarities. To align the temporal evolution as well as possible, an optional time shift of up to 10% is permitted in the aggregated similarity computation. This led to a computation time of 137 s for the distance matrix D_R.

One obtains the dendrogram shown in Figure 4.16c using complete linkage. In the first analysis step, a pruning level leading to two clusters is selected. The hyper-slicer reveals small values for the parameters *pulse delay* and *exciteOverGap*. Thus, these parameters significantly influence the simulation outcome, which confirms the hypothesis of the domain experts that the other two parameters are less influential.

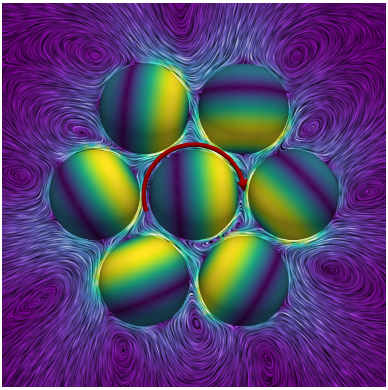
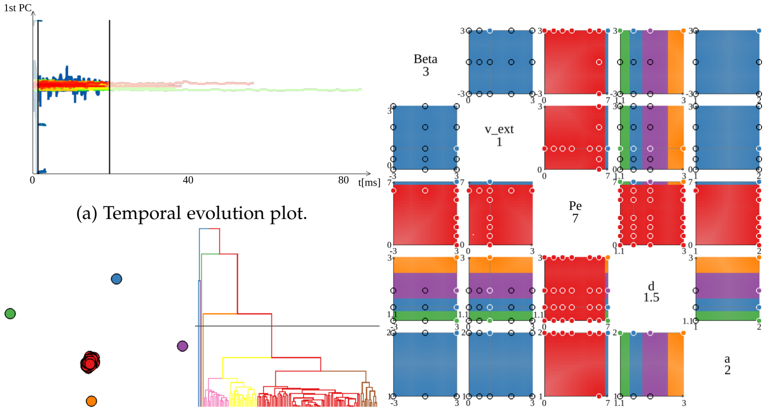


Figure 4.17: A system of seven active particles shows a complex motion due to the hydrodynamic interactions between the particles. One interesting observation is a rotation of the central particle, indicated by the red arrow.

Next, the domain expert refines the cluster as shown in Figure 4.16. The corresponding parameter-space partitioning embedding reveals that not all clusters found in similarity space are also connected in parameter space. For example, the orange cluster is separated into multiple segments. Additionally, one can identify the green and the red clusters as being the largest. The domain expert is especially interested in the purple cluster. After selecting it, its boundaries from outside the plane can be projected in the hyper-slicer. The projection shown in Figure 4.16d reveals that this segment only occurs for small values for the parameter *pulse delay* and the parameter *exciteOverGap* lies between 3 and 9. Thus, the cluster is located in a clear region in the parameter space, even though the segmentation embedding reveals that the cluster is split into two segments. When selecting single ensemble members and visualizing them in the slice viewer, the domain expert observes two separate peaks with a small connection (see Figure 4.16b) that he identifies as characteristic of this cluster. Our visualizations allow for connecting this characteristic behavior back to the input parameter settings.

4.8.3 *Active Crystal*

Active particles are particles that can use energy from their surrounding to propel themselves [28]. For this use case, we consider active crystallites, which are small active crystals where each particle is fixed in its position but can rotate freely [101]. The active crystallites investigated here



(a) Temporal evolution plot.

(b) Similarity embedding.

(c) Clustering dendrogram.

(d) Hyper-slicer.

Figure 4.18: The time interval of interest for the microswimmer dataset is selected in the temporal evolution plot (a). The similarity embedding shows five similarity clusters (b) that can be chosen as separate clusters in the clustering dendrogram (c). The induced parameter-space partitioning can then be analyzed in the hyper-slicer (d).

comprise seven particles forming a hexagon with one particle in the center, see Figure 4.17 for an example. The particles can either be spherical or ellipsoidal, which allows the shape of the particle to be described by an aspect ratio *a*. The particles, also referred to as microswimmers, perform a swimming motion. The propulsion mechanism is modeled by the so-called squirmer model, driven by a parameter *beta* [208]. Additionally, the particles may be influenced by Brownian motion. The strength of the influence compared to the propulsion speed is described by the Péclet number Pe. The particles create a flow field by their self-propulsion, leading to a complex interaction between them. However, these simulations also allow for studying the influence of an additional external flow field with a velocity magnitude *v_ext* surrounding the active crystallite. This dataset aims to understand the particles' complex motion and find different kinds of behavior. This leads to different states of matter whose extent in parameter space is of interest.

The originally simulated data consists of a flow field for the velocity of the surrounding fluid as well as a scalar field that describes the pressure. Additionally, the orientations for each particle are stored in each step leading to a set of time-varying orientation vectors. The simulation outcome depends on the five different driving parameters described above.

In the following, we use our approach to understand the parameter dependency of the self-organization of small swimmers. In this chapter, we investigated the pressure field of the flow simulation as the simulation output, which depends on the five different input parameters. The parameter Pe is studied on the logarithmic scale for a more meaningful analysis. The ensemble contains 115 runs with 18 to 308 adaptive time steps each. The scalar fields have a resolution of $128 \times 128 \times 64$.

The temporal evolution of the ensemble is shown in Figure 4.18a. We immediately observe that the runs vary in length, and there is little variation in the latter time steps for those runs that cover a long time. Additionally, we observe a short transition phase at the beginning of the simulation. Therefore, the time interval for the following analysis is limited from 1.3 ms to 19.7 ms. This time interval was used to compute the aggregated distance matrix D_R in 17.8 s.

The similarity embedding created based on this matrix D_R is shown in Figure 4.18b and reveals five clusters. Using *ward.D2*, we obtain a hierarchical clustering as shown in the dendrogram in Figure 4.18c. In the dendrogram, we select a pruning level such that the clustering matches the clusters in the similarity embedding. In this clustering, four of the clusters only consist of single runs. These correspond to runs that strongly differ from all the others and can be considered outliers. The hyper-slicer (see Figure 4.18d) shows the locations of these runs in parameter space. They all share the parameters of *beta*= 3, *v_ext*= 1, *a*= 2 and also contain the same Péclet number. Thus, the distance between the particles is the only parameter that separates these runs. Therefore, our approach allowed for finding a parameter region where the simulation runs significantly differ from all the others. This region would benefit from a denser sampling.

4.9 DISCUSSION

We collected feedback from all four domain experts that explored their data with our tool. They consider the tool very helpful and acknowledge that it facilitates understanding the parameter's influence on the data. The importance of linking abstract visualizations to the explicit spatial visualizations of individual ensemble members was stressed by two domain experts.

The domain expert who created the blood flow dataset obtained new insights into the importance of the different parameters. Our approach also confirmed the prior hypothesis about the strong influence of the boundary conditions and the characteristic length. The other domain expert from the medical domain expressed interest in including her experimental data in the visualization. This would allow her to find the parameter values that fit best the measured data.

The physicist who works on microswimmers mentions that he sees new possibilities to analyze data with more than three parameters which currently imposes large challenges in his domain. This usually leads him to limit his analysis to fewer parameters, even though including all of them might yield additional insights. The limitation in the number of parameters might even lead to missing important phenomena. The hyper-slicer supported him in obtaining a mental picture of the parameter space and its partitioning, so he rated this visualization as helpful. He also positively highlighted the high information density.

The physicist who created the semiconductor simulations stated that our analysis approach shortens the time for the analysis process. Additionally, it pointed him toward parameter dependencies that he would like to investigate further by creating additional simulations. Before exploring his data with our tool, he was unaware of this region of interest in the parameter space. He sees a potential application of our approach in identifying sub-spaces of the parameter space that are then used for a more detailed analysis.

Overall, three of the domain experts highlighted our approach's broad applicability. However, they also saw some limitations and proposed valuable suggestions for improvement. One of them desired more visual support in investigating selected clusters. The uncertainty visualization was also included after it was proposed by the first domain expert whom we asked for feedback. This was then included before the sessions with the other domain scientists. While the users rated the interactive exploration as intuitive and helpful, it became clear that initial training is necessary to work with our tool efficiently. However, as this is the case for most more complex analysis tools, the need for training does not reduce the intuitiveness of users who received the training.

In summary, the approach presented in this chapter allows for the interactive visual analysis of simulation ensembles and their parameter spaces. The approach supports creating and analyzing parameter-space partitionings on different levels of detail. It received overall positive feedback from the domain experts, allowing them to confirm existing hypotheses and gaining new insights into their data.

Even though the approach enables the analysis of multi-dimensional parameter spaces, the hyper-slicer does not scale well with an increase in the dimensionality, and a manual selection, even if supported by an ordering, might not remain feasible for very high-dimensional parameter spaces. However, densely sampling many dimensions in the parameter space is barely possible for spatio-temporal simulation ensembles due to the high computational cost of running the simulations. Therefore, we do not see a (practical) limitation in the limited scalability with the number of dimensions.

One of the domain experts suggested selecting more than one cluster at once for comparative analysis. While a generalization to several clusters is straightforward, the visualizations quickly become cluttered. Therefore, finding a solution to this problem is a topic for future research. Additionally, a more structured user study for investigating the perception and navigation through multi-dimensional spaces would be beneficial. This could also include a comparative evaluation of the parameter-space overview visualizations. Since the segmentation embedding was added after collecting feedback from the domain experts, it is unclear if they rate this visualization as more intuitive and beneficial than the MDS embedding of the parameter-space samples.

5

INTERACTIVE DEFINITION OF CHARACTERISTIC MEASURES

The results of numerical simulations can often be summarized by particular characteristics, which allow for reducing the dimensionality of the data. However, the definition of these characteristic measures strongly depends on the data and analysis tasks. At the same time, defining these measures is one of the core tasks in data analysis and can become very challenging. Therefore, in this chapter, we propose an interactive tool for the definition of characteristic measures.

One application area where the definition of characteristic measures is particularly important are systems of active particles as introduced in Section 4.8.3. In contrast to the previous chapter, we do not investigate the scalar fields but instead focus on the orientation of the particles. In this field of research, the characteristic measures commonly correspond to order parameters which allow for differentiating between different states of matter. In general, we consider a small active crystal consisting of k active particles which are fixed in their positions but can rotate freely. Even though the fixation does not allow translational motion, the self-propulsion of the particles creates a complex hydrodynamic interaction which leads to rotations of the particles. Each of the individual particles can be described by a three-dimensional direction vector which describes the swimming direction of the particle. Thus, the whole system for a single simulation run can be described by a time-varying $3k$-dimensional feature vector created by concatenating the feature vectors of the individual particles. This results in the challenge of analyzing a multi-dimensional feature vector varying over time. Defining characteristic measures for this type of data allows for facilitating its visualization. It also helps to differentiate between simulation runs for different parameter settings and compare them concerning specific characteristics. While many measures have been proposed for the analysis of 1D time series, it is unclear at the beginning of the analysis process which measure is suitable for the analysis in a multi-dimensional domain.

For a comprehensive analysis, it is important to define the measures on two different levels of detail. First, the evolution over time should be represented by the measure. Later, it is of interest to find a measure that also aggregates over time and, thus, leads to a single scalar value for each ensemble member. This significantly facilitates the analysis of, for example, the dependency on input parameters.

In this chapter, we present a requirement analysis and task abstraction from the domain of active systems. Based on this analysis, we propose a workflow for interactively defining characteristic measures. We also present ASEVis (Active System Ensemble Visualization), an interactive visual analysis tool that implements the identified workflow. Its utility is shown by presenting a use case where the temporal dynamics of active systems are analyzed.

After providing an overview of the related work in this field (see Section 5.1), we present a requirement analysis together with the task abstraction in Section 5.2. Section 5.3 described the workflow for defining active measures, followed by the presentation of our visual analysis system in Section 5.4. Finally, we present a use case for analyzing active systems in Section 5.5.

The work presented in this chapter has been published in the following paper:

> **M. Evers**, R. Wittkowski and L. Linsen, ASEVis: Visual Exploration of Active System Ensembles to Define Characteristic Measures, *2022 IEEE Visualization and Visual Analytics (VIS)*, 150-154 (2022)

I implemented the approach and created the results. Raphael Wittkowski provided feedback from the domain expert's perspective. All authors contributed to editing the manuscript.

5.1 RELATED WORK

The definition of new measures in science is commonly performed using general-purpose programming tools like Jupyter notebooks [192] or scripting interfaces to programming tools like programmable filters in Paraview [4]. However, these approaches are limited in using interaction. While the Javascript-based Observable notebooks [249] overcome this limitation, they do not provide access to many of the standard data processing libraries that are available in Python. Additionally, the general-purpose notebooks do not offer visualizations that can be used directly and target the problem at hand.

Different visualization approaches target user-defined feature definition in multi-dimensional data [1, 41, 118, 323, 165]. Liu et al. [212, 211] propose visualization tools where users can define features by sketching. While sketching is easy for the user, it does not provide the large flexibility that comes with programming languages, which also allow defining features and measures derived from the data at hand. Jänicke et al. [178] use methods from information visualization to identify features. Cellpackexplorer [305] neither targets the definition of derived measures nor features but provides a visual aid for model building. Closest to our approach is Paraglide [34] which supports the analysis of simulation data on its input parameters. The authors state that constructing derived measures is one of the requirements in their system, and it can be closely integrated into different programming environments. However, it does not allow working with time-dependent data, which adds additional challenges.

Many methods deal with the analysis of trajectory data, which can be seen as multi-dimensional time-series, more generally [15]. Using predefined features for the analysis [241, 247] assumes that the definition of the feature is known before the analysis. Luboschik et al. [220, 221] also make this assumption in their tools targeted at studying the influence of parameters. While Zhao et al. [398] allow for creating derived time series interactively, they do not support aggregations. While most of the approaches focus on the analysis of 2D or 3D trajectories [73, 146], Amirkhanov et al. [12] recently proposed a generalization to 4D trajectories. Wulms et al. [384] present a static visualization that reduces the dimensionality of the trajectory data to only a single dimension.

5.2 REQUIREMENT AND TASK ANALYSIS

In this chapter, we analyze the motion of an active crystallite (a small crystal) as introduced in Section 4.8.3. As we are only interested in characterizing the temporal patterns in the system, we only consider the particle directions and do not analyze, for example, the flow field surrounding the particles. The particles' motion considered in this chapter is characterized by the propulsion mechanism (driven by the parameter `beta`) as well as by the distance between the particles (parameter `d`) while we keep the other parameters discussed in the previous chapter fixed. While the state of every single particle can be described by a three-dimensional particle direction vector, the state of the whole system is given by a multi-dimensional feature vector that is formed by concatenating the individual particle direction vectors as shown schematically in Figure 5.1. In the case of 7 particles, this leads to a 21-dimensional feature vector. Thus, the

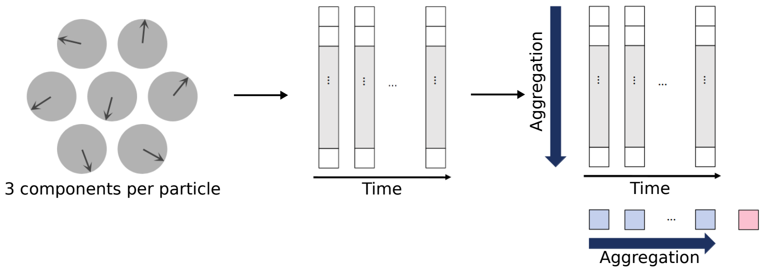


Figure 5.1: Describing the dynamics of an active crystal: The motion of each particle is described by a 3D direction vector. Concatenating these vectors leads to a multi-dimensional feature vector for each time step.

data can be described as an ensemble of multi-dimensional time series or trajectories in the 21-dimensional space spanned by the different components of the feature vector.

A common way to interactively define derived measures in physics data is using Python scripts or Jupyter notebooks. However, as both approaches commonly lead to static plots, an interactive exploration is barely possible. Instead, even simple operations like zooming require a recreation of the visualizations. Thus, understanding the properties of the data leads to many recreations of the graphs. This interrupts the workflow frequently, especially when finding new measures for aggregation is one of the core tasks of the analysis. Based on our analysis for studying the temporal behavior of this system [101], we identify a set of requirements:

R5.1 Visualizations of the time series on different levels of detail shall provide an understanding of the dataset.

R5.2 The users shall be able to define an aggregation measure over time to reduce the dimensionality of the data. Simpler visualizations over time based on this measure support a comparison of ensemble members and the selection of time intervals of interest which might also involve skipping transition phases.

R5.3 The definition of an aggregation measure over time shall aggregate each ensemble member to a single scalar value. This aggregation allows a comparison over the whole ensemble and creates a relation to the input parameters. Visualizing the parameter space using the aggregated measure allows for investigating different states of the active system.

These requirements from the domain of active systems can be abstracted to tasks using terms commonly used in visualization research. The tasks support an easy generalization to other domains dealing with multi-dimensional time series:

T5.1 Visualization for individual multi-dimensional time series (R5.1).

T5.2 Allow the users to interactively specify how to aggregate the values of a multi-dimensional time series into a single scalar value for each timestep (R5.2).

T5.3 Visualization for the aggregated measures over time (R5.2).

T5.4 Allow the users to interactively specify how to aggregate the time series to a single scalar value for each ensemble member (R5.3).

T5.5 Visualization of the parameter space using aggregated data values (R5.3).

We will explain the workflow and the visual designs to address the tasks and requirements in the following.

5.3 WORKFLOW

Based on the previously defined requirements and task, we define an interactive visual analysis process that is shown schematically in Figure 5.2. The analysis process starts by an iterative bottom-up procedure to define the aggregation measures per time-step as well as over time. When investigating single ensemble members in detail (T5.1), the users obtain an initial understanding of the data. By using an interactive Python programming interface that is directly integrated in the visualization tool, the users can define an aggregation for each time step (T5.2). This user-defined measure can be observed in a visualization over time (T5.3). If necessary, the definition of the aggregation can be refined. Otherwise, an aggregation over time can be interactively defined by using the interactive programming interface again (T5.4). The last step of the measure definition process is the observation of the aggregation (T5.5) which might induce additional iterative refinement steps.

The bottom-up measure definition step is followed by a top-down visual analysis of the whole ensemble to obtain deeper insights into the data. In a first step, the distribution of the aggregated values can be observed in dependence of the parameter values (T5.5). Here, it is possible to select ensemble runs of interest which could then be visualized over time (T5.3). As the visualization over time still builds on aggregated data

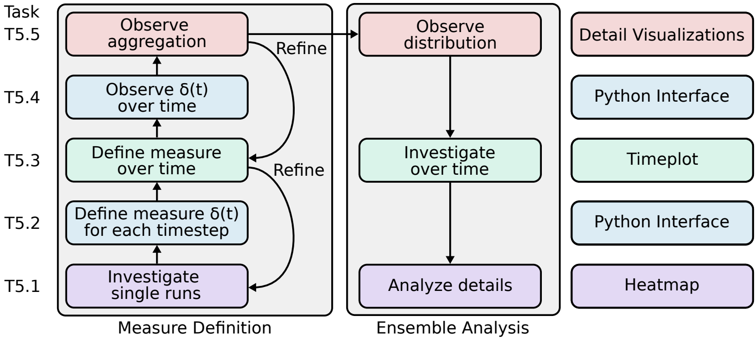


Figure 5.2: The interactive workflow starts with a bottom-up analysis where time-dependent and time-independent measures are iteratively defined. These measures are then used for an analysis of the ensemble data.

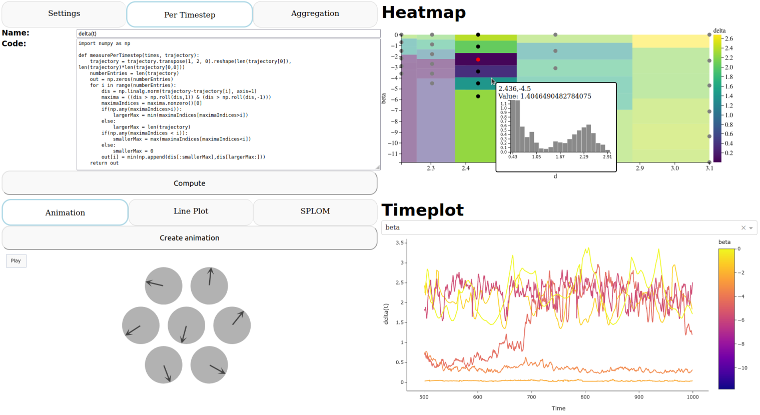
for each time step, the users can select single ensemble members for an investigation on the highest level of detail to understand the behavior of the simulation run (T5.1).

5.4 VISUAL ANALYSIS SYSTEM

To implement this workflow, we design an interactive visual analysis system *ASEVis* containing visualizations on three different levels of detail. A screenshot of the tool *ASEVis* is shown in Figure 5.3. For the detail visualizations, we combine an animation with a line plot showing the evolution over time and a scatterplot matrix where we used PCA to reduce the dimensionality of the originally 21-dimensional space. For visualizing the aggregations over time, we propose to use a line plot which we refer to as a *timeplot* in the following. The most aggregated data can be visualized in a heatmap spanned by the two parameter values. In addition to the visualizations, the analysis tool contains an interactive programming interface that allows for defining measures and is closely linked to the different visualizations.

5.4.1 *Detail Visualizations*

Three detailed visualizations allow for understanding the behavior of single runs, which forms the base for defining the measure and also supports the analysis on the highest level of detail (R5.1). An *animation* of the particle orientations provides a descriptive understanding of the ensem-



Name: delta(t)

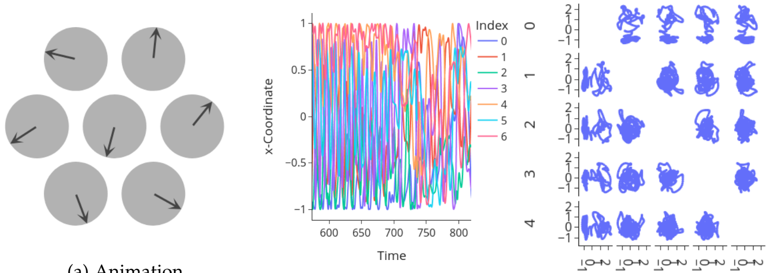
Code:

```
import numpy as np

def measurePerTimestep(times, trajectory):
    trajectory = trajectory.transpose(1, 2, 0).reshape(len(trajectory[0]),
len(trajectory)*len(trajectory[0,0]))
    numberEntries = len(trajectory)
    out = np.zeros(numberEntries)
    for i in range(numberEntries):
        dis = np.linalg.norm(trajectory-trajectory[i], axis=1)
        maxima = ((dis > np.roll(dis,1)) & (dis > np.roll(dis,-1)))
        maximaIndices = maxima.nonzero()[0]
        if(np.any(maximaIndices>i)):
            largerMax = min(maximaIndices[maximaIndices>i])
        else:
            largerMax = len(trajectory)
        if(np.any(maximaIndices < i)):
            smallerMax = max(maximaIndices[maximaIndices<i])
        else:
            smallerMax = 0
        out[i] = min(np.append(dis[:smallerMax],dis[largerMax:]))
    return out
```

Figure 5.3: ASEVis is a tool supporting the visual analysis of active systems and allows interactively defining characteristic measures.

ble run and how the particles move, see Figure 5.4a for a static image taken from the animation. However, long animations or many animations are cognitively demanding compared to static visualizations [123]. Therefore, we also include a static visualization of single components of the particle orientation vector over time. In a *line plot* with time on the horizontal axis, we visualize either the x-, y- or z-component of the particle direction vector where we color-code the individual particles. Our system with seven particles results in seven lines, as shown in Figure 5.4b. As we leave out two components of each orientation vector, this visualization does not include the whole data but allows for identifying important features in the data more easily than the animation. We use a *scatterplot matrix* (SPLOM) to include all multi-dimensional particle vector components. However, the 21 dimensions of the dataset lead to a relatively high dimensionality which requires a lot of screen space and is hard to interpret. Therefore, we first perform a principle component analysis (PCA) [177], which allows for extracting the most important features and neglecting dimensions that do not contribute. This leads to a visualization as shown in Figure 5.4c. Here, we select the principal components that cover 99.9% of the variation. To limit the dimensionality in case of many dimensions necessary, the user can optionally set an upper limit of dimensions to include. The result of the PCA is then visualized in the SPLOM. Besides including all components in the analysis, this vi-



(a) Animation

(b) Lineplot

(c) SPLOM

Figure 5.4: Three different detail visualizations allow for investigating single ensemble members. The line plot for $d = 2.436$ and $\beta = -4.5$ reveals that until $700R/B_1$, the particles change their orientation almost periodically, while for larger times, it is completely aperiodic.

sualization allows us to identify the intrinsic dimensionality of the data, which might be helpful for a potential measure definition.

5.4.2 *Interactive Programming Interface*

We include *interactive programming interfaces* for defining measures to aggregate data per time step and over time (T5.2, T5.4). This interface allows users to directly define suitable measures during the analysis process. Integration directly into the analysis tool also facilitates the explorative analysis based on the definition of the aggregation measure as well as iterative refinement. As Python is the most common programming language in the application domain, we also use it for our interface. Additionally, it allows for using a wide range of standard libraries like `numpy` and `scipy`.

For defining the measures, the users implement a function with a specific signature provided in a function template as a starting point. As both measures serve the very specific purpose of reducing the dimensionality of a multi-dimensional vector to a scalar for each time step and the time series to a single scalar, the need to follow the signature does not impose a restriction. Additionally, it is possible to use the aggregation measure per time step for defining the measure aggregating over time which facilitates the development. Users can name each measure which is then used to label the corresponding visualizations.

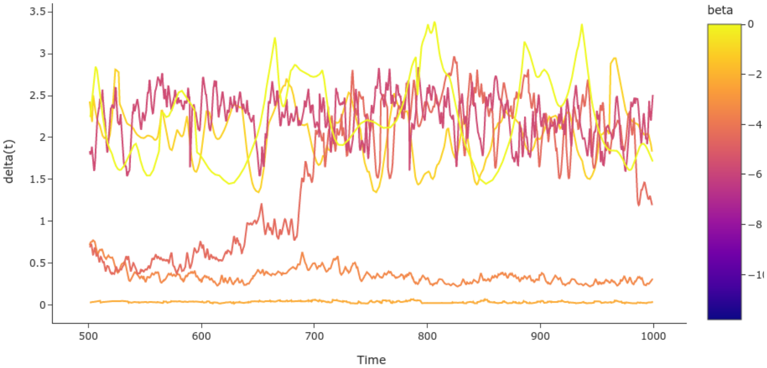


Figure 5.5: The timeplot shows the variation of the aggregation for every single timestep over time. The lines can be color-coded according to the input parameters to study their influence.

5.4.3 *Timeplot*

A line plot that shows the user-defined measure over time is included to visually compare the different ensemble members over time (R5.3) as shown in Figure 5.5. We refer to this plot as a *timeplot* to differentiate it from the detail visualizations. The measure is evaluated for each time step individually. To differentiate the different ensemble members, the users can select among a color coding highlighting only the different runs or encoding the parameter values by color. This allows understanding of how different input parameters influence the variation over time. The color coding can be changed interactively to better understand the parameter values.

5.4.4 *Heatmap*

A *heatmap* allows for visualizing the aggregations over time for each ensemble member. For the axes, we choose the two parameters. The heatmap resembles a state diagram which is a typical representation in studying the occurrence of different states based on the parameter values. We use the `viridis` [350] color map, which is perceptually uniform. For irregularly sampled parameter spaces, the segments for the sample points are extended along the axes to cover the whole parameter space and allow for an easier interpreation compared to showing only points. This might lead to segments of the heatmap where the sample point is

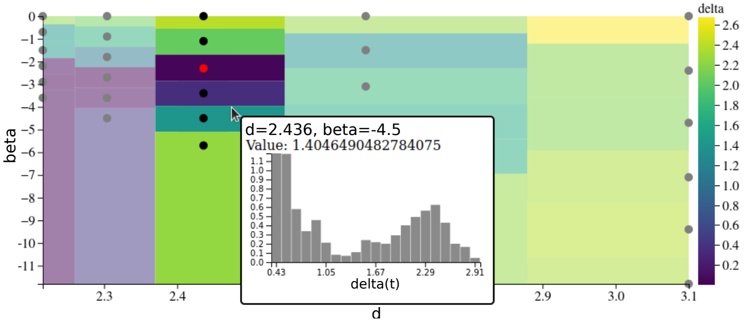


Figure 5.6: A heatmap visualization shows the aggregation over time for each ensemble member depending on the parameter values. When hovering over fields, a histogram shows the distribution of the values per timestep. Ensemble members can be selected interactively for investigation in linked visualizations.

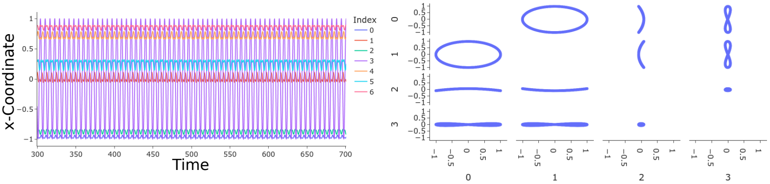
not placed in the center. To clearly show the sampling of the parameter space, the location of each sample is marked by a black point.

The users can interactively select ensemble members by brushing in the heatmap. Selected regions are visualized by changing the color of the sample point to red which is chosen as it does not interfere with the colormap used for the heatmap. Additionally, the saturation of unselected cells is decreased to 0.5. As the visualizations are closely linked, a set of selected ensemble members can be directly shown in the timeplot. It is also possible to select individual ensemble members, which are visualized in the detail visualization.

To validate the choice of the aggregation measure over time, we also show the distribution of the aggregation measure per time step. This distribution is shown in a histogram when hovering over a cell on the heatmap. It also provides additional information in the analysis stage that goes beyond the aggregation to single scalar values but does not include the complexity when visualizing the data over time.

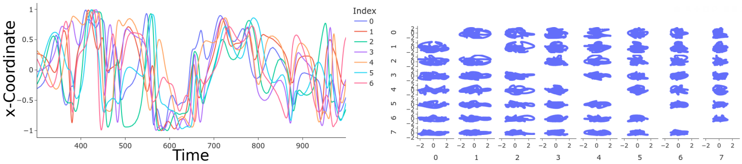
5.5 ANALYZING ACTIVE CRYSTAL DYNAMICS

In this application use case, we will describe how our tool was used for defining an order parameter in a study of the properties of active crystals [101]. One of the project's main goals was to define an order parameter that differentiates between different temporal states. Then, it should be studied for which of the parameter values d, which describes the distance between the particles, and β, which characterizes the propulsion



(a) Line plot for periodic case

(b) SPLOM for periodic case



(c) Line Plot for aperiodic case

(d) SPLOM for aperiodic case

Figure 5.7: Detail visualizations allow for investigating single ensemble members. We can differentiate between periodic behavior (top row, $\beta = -2.7$ and $d = 2.3R$) and aperiodic behavior (bottom row, $\beta = 0.0, d = 2.3R$).

mechanism, these different states occur. Especially boundaries in parameter space and transitions between the different states are interesting. For the analysis, we used a dataset that describe the system that we also analyzed in Section 4.8.3 but only consider the orientations of the particles. The dataset consists of 30 runs. Each simulation run covers a time of $1000R/B_1$ (dimensionless time units), which results in 465 to 6664 time steps.

We follow the analysis workflow presented in Section 5.3. At first, we analyze individual ensemble members. In Figure 5.7, we spot very different behavior between periodic states (top) and aperiodic states (bottom). A first investigation revealed a transition phase which is why, in the following, we only consider the simulation results starting at $300R/B_1$. While the line plot shows the difference between the ensemble members, it only visualizes one of the orientation vector components. Observing the behavior in a SPLOM reveals that the ensemble member with $\beta = -2.7$ and $d = 2.3R$ indeed shows periodic behavior and only the first four principal components are needed for covering 99.9% of the variation in the data. In contrast, the run with parameter values $\beta = 0.0$ and $d = 2.3R$ does not show any periodicity in the line plot shown in Figure 5.7c. Additionally, the PCA reveals that we would need the first 20 components to cover 99.9% of the variation. To avoid too many scatterplots for the available screen space, we limit the number of components included in the SPLOM to 8, as shown in Figure 5.7d. Also, in this visualization, we cannot identify any periodicity. Therefore, we aim to define a measure for differentiating periodic and non-periodic behavior.

Based on insights obtained from studying individual ensemble members, we defined a suitable measure interactively. At first, we experimented with the distance to the first timestep. While this measure would become 0 regularly for periodic behavior and, thus, the resulting one-dimensional time series covers the periodicity, it does not yield meaningful results for aperiodic cases. Additionally, it does not directly support the identification of the end of the transition phase and strongly depends on the choice of the first time step. After several iterative refinement steps, we identified a measure that satisfies our requirements. Here, we use the closest Euclidean distance to a previous position where the multi-dimensional trajectory returns to. This can be described as

$$\delta(t) = \min\Big(\min_{0<r<m_1} \|\mathbf{u}(t) - \mathbf{u}(r)\| \ , \\ \min_{m_2<r<t_{\max}} \|\mathbf{u}(t) - \mathbf{u}(r)\| \Big) ,$$

where $\mathbf{u}(t)$ describes the 21-dimensional feature vector at time t, m_1 is the closest local maximum of the function $f(r) = \|\mathbf{u}(t) - \mathbf{u}(r)\|$ with $m_1 < t$ and m_2 is the closest local maximum of $f(t)$ with $m_2 > t$ [101]. The timeplot shown in Figure 5.5 reveals a clear separation between the different behaviors. For periodic and static cases, the measure $\delta(t)$ decreases to 0 while $\delta(t) > 0$ holds for all values of t if the ensemble member is aperiodic. Additionally, this behavior allows for clearly defining the end of the transition phase by observing when it decreases to 0 for periodic runs.

To aggregate the data over time and obtain a single scalar value characterizing the ensemble member (R5.3) we use the mean value of the selected time interval. This aggregation results in a heatmap as shown in Figure 5.6 where we can observe different regions. Observing the results shows that periodic behavior occurs for small values of d, and the defined measure increases continuously in the direction of d. Observing the changes in the direction of β, we can find discontinuities indicating a different kind of transition between the corresponding states.

For a deeper analysis of the discontinuous transition, we select a set of simulation runs with $d = 2.436$ and different values for β. The selected simulation runs are shown in the timeplot in Figure 5.5. Using the value of β as a color coding, we can observe that the ensemble members with $\beta = -2.3$ and $\beta = -3.4$ show values of $\delta(t)$ that are close to zero while the values for the other members are higher. The only exception is the temporal evolution of $\beta = -4.5$ which shows small values until $t = 700R/B_1$ and, at this point, suddenly jumps to higher values.

To understand this phenomenon, we can select this ensemble member and investigate its behavior in the animation as well as the other detail visualizations. The line plot of the x-components of the particles' direction vectors is shown in Figure 5.4b. This visualization, as well as the animation, reveals that initially, the motion seems almost periodic. At later timescales, however, the motion appears chaotic where we can identify the transition around $t = 700R/B_1$, which agrees with the steep increase in the timeplot.

5.6 DISCUSSION

The visual analytics approach presented in this chapter supports the interactive definition of characteristic measures in the domain of active matter physics. Based on a requirement analysis, we defined a set of abstract tasks. These tasks were addressed in a visual analysis tool[1] that supports the analysis of complex active systems. Our approach is helpful in the physics domain because the order parameter defined with the help of this approach led to a physics article [101].

Our approach generally scales well to larger systems as it builds on a user-defined dimensionality reduction. If the number of particles grows significantly, it might be necessary to adapt the detail visualizations to avoid overplotting. Additionally, we only considered two-dimensional parameter spaces in this chapter as this was sufficient for the presented application use case. However, exchanging the heatmap for a suitable multi-dimensional visualization does directly allow for applying this approach to higher-dimensional parameter spaces. One possible multi-dimensional parameter-space visualization here would be the hyperslicer as presented in Chapter 4.5.1.

While the findings generated by this approach resulted in a domain-specific article and, thus, the tool also proved helpful in practice, its broader applicability needs to be investigated in more detail. In general, the approach presented in this chapter can be easily generalized to other domains in which simulation ensembles, where a multi-dimensional feature vector can describe each ensemble member, are analyzed. Only the detail visualizations are domain-specific and need to be exchanged, while all other visualizations could be applied directly. Future research might include the generalization and easy applicability to other data types, requiring a flexible approach for the detail visualizations. Similar measures are also needed for spatial datasets. An interesting research direction would be to study the application of this approach to feature

---

[1] `https://github.com/marinaevers/asevis`

vectors derived by spatial simulation ensembles and how to include the selection of suitable feature vectors directly in the analysis workflow. One possibility to use spatial data directly is using the feature vectors created by Monte Carlo sampling as introduced for the multi-run similarity plot (see Section 2.3.2).

Additionally, the adaption by domain experts would be significantly improved by providing more options to create paper-ready figures. In this chapter, we only studied simulation ensembles that can be defined by a multi-dimensional feature vector for each ensemble member. Also, the workflow for defining characteristic measures as presented in this approach could be combined with the analysis of parameter-space partitionings as presented in Chapter 4 where a feature-specific parameter space partitioning could be computed from the aggregated data obtained by the user-defined dimensionality reduction.

6

TOPOLOGICAL ANALYSIS OF PARAMETER DEPENDENCIES

The topological structure of scalar fields in any dimension is commonly used to analyze intrinsic features of the data. The fields' topology is a powerful descriptor of the scalar field because the topological structure is invariant under continuous deformations. Therefore, topological data analysis has been increasingly applied to study ensemble data in recent years. One application is understanding the input parameter's influence on the data's topological structure. Therefore, it is interesting to study the topological variations in dependence on the parameter value.

Common topological descriptors for scalar fields are the merge and the split tree. They can be combined into a contour tree which encodes the topology depending on the isovalue. However, since visualizations of the contour tree as vertical tree visualizations or node-link diagrams in the spatial domain are non-intuitive to non-experts, Weber et al. [367] proposed the visual metaphor of topological landscapes. Topological landscapes are 2D scalar fields described by the same contour tree as the multi-dimensional scalar field. Thus, they can be used as 2D representation of the scalar field's topology.

However, if topological landscapes should be used to study topology variations depending on simulation parameters, the topological landscapes are not coherent, hindering interpretability. Therefore, we propose a concept for creating coherent topological landscapes based on finding mappings of topological features in merge and split trees that can be combined to contour trees. The resulting coherent contour trees over a selected input parameter can be visualized as animations of coherent topological landscapes or 3D volume visualizations of stacked topological landscapes. This chapter is based on an extended version of a paper by Herick et al. [154] that describes temporally coherent topological landscapes. As time can be interpreted as a particular case of a parameter, the original approach is included in the work presented in this chapter. Therefore, for the remainder of this chapter, the temporal variation is included when discussing variations over a parameter.

In the following, we will summarize related work in topological ensemble analysis and discuss the necessary background for this chapter in Section 6.1. After providing an overview of our approach (see Section 6.2), we present our method for computing coherent contour trees in Section 6.3. The visualization of coherent topological landscapes is presented in Section 6.4 followed by different application use cases in Section 6.5.

The results presented in this chapter are based on the following paper:

> **M. Evers**, M. Herick, V. Molchanov and L. Linsen, Coherent Topological Landscapes for Simulation Ensembles, In *Computer Vision, Imaging and Computer Graphics Theory and Applications*, edited by Bouatouch K. et al., 223-237 (2022)

Maria Herick developed and implemented the method for coherent topological landscapes and applied it to temporal data as described in the conference paper [154]. All authors discussed the methods of the extended paper and contributed to writing and editing the manuscript of the paper. I implemented the generalization to simulation ensembles and created the results.

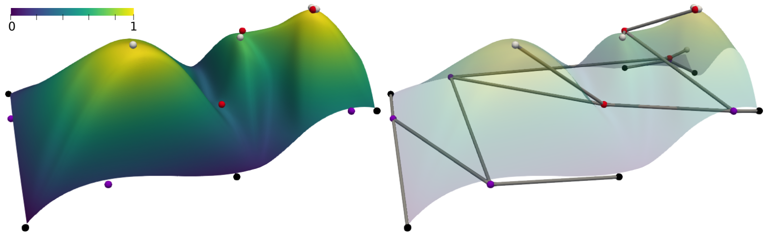
6.1 RELATED WORK AND BACKGROUND

Different topological descriptors for scalar fields have been proposed, for example, the analysis of persistent homology that can be shown as persistence diagrams or barcode visualizations [89]. The Morse-Smale complex partitions the domain so that each domain is equivalent in gradient behavior. [132, 131].

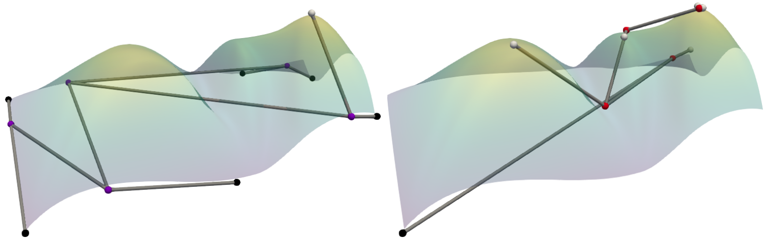
A level set L_z [90] for a value $z \in \mathbb{R}$ for a fixed parameter value p_i is defined as

$$L_z(f) = \{x | f(x, p_i) = z),$$

where $f(x, p_i)$ is a Morse function. In the case of 2D data, a simply connected component of L_z is called an isoline, and in 3D data, it is called an isocontour. A Reeb graph [282] can then describe the evolution of the level sets when varying the isovalue z. If the function f is defined on a simply connected domain, the general Reeb graph simplifies to a contour tree [351, 56, 83]. The contour tree nodes represent the function's critical points f. The leaves of the tree describe the emergence of new isocontours (minima) or the vanishing of a contour (maxima). The internal nodes represent saddle points at which isocontours merge or split.



(a) Data. (b) Contour tree.



(c) Merge tree. (d) Split tree.

Figure 6.1: The scalar field (a) is shown as a height field together with its critical points (black: minima, violet: saddle points with merge, red: saddle points with split, white: maxima). The contour tree (b) is constructed by combining the merge tree (c) and the split tree (d). All trees are visualized with the nodes located on their positions in the spatial domain and the corresponding scalar field shown as desaturated surfaces.

Based on these two options, the saddle points can be classified as merge or split nodes. An edge of the contour tree represents the lifespan of the topological feature corresponding to the nodes that the edge connects. Thus, each point on the edges of the contour tree represents one isocontour, and each cut through the contour tree at a certain height represents a level set.

An efficient computation for contour trees has been proposed by Carr et al. [56]. As intermediate steps, a merge tree and a split tree are created. The merge tree, also called join tree, contains all minima, all merge nodes, and the global maximum. The split tree is formed by the maxima, the split nodes, and the global minimum. Both split and merge trees are rooted trees. Merge and split trees can then be combined to obtain the contour tree, which is an unrooted tree. An example of the topological structure of a simple dataset is shown in Figure 6.1. Here, we explicitly show the dataset together with the critical points as well as the split tree, the merge tree, and the resulting contour tree. Many options for the computation of topological features are included in the Topology

ToolKit (TTK) [340, 339, 36], which we also used for the computation and simplification of split and merge trees.

Contour trees can be visualized directly using a 2D graph drawing algorithm to show them as a node-link diagram. The vertical axis can show the corresponding scalar values [56]. Alternatively, if the domain of the scalar field is 2D or 3D, the nodes can be shown directly in the spatial domain as shown in Figure 6.1. Pascucci et al. [255] proposed using a 3D radial graph drawing algorithm which has been extended to a visualization called "topological cacti" [368]. However, analyzing the topological structure of the data in node-link diagrams requires a significant amount of background knowledge in creating contour trees. Therefore, Weber et al. [367] proposed topological landscapes as a terrain metaphor that encodes the contour tree.

6.1.1 *Topological Landscapes*

Topological landscapes [367] can be used to create a 2D heightfield that contains the same contour tree as the original data, which can be of arbitrary dimension. As the landscape is described by the same contour tree as the original data, it contains the same topological information, but it is easier to interpret as a direct tree visualization of the contour tree. When constructing a topological landscape, the position of the individual peaks and their areas are not determined. Therefore, these attributes can be used for encoding additional information, like the volume of the topological features.

The computation of topological landscapes is based on computing the so-called branch decomposition, an acyclic graph consisting of monotonous paths of the contour tree. Thus, the nodes along a path need to increase or decrease monotonously. The graph created by a branch decomposition can be seen as a hierarchical graph where a branch b is a child of another branch $\hat{b}$ if one of b's endpoints is one of $\hat{b}$'s inner nodes.

This hierarchical structure can be used to iteratively construct the topological landscape as shown schematically in Figure 6.2. The root branch (gray in Figure 6.2) corresponds to the entire domain of the landscape, which is stored as a triangulation. The root branch's minimum (black) is assigned to the boundary vertices, while the maximum (white) is assigned to the central vertex. Before the children can be added, the triangulation has to be refined to create enough space. Then, the extremum corresponding to each child branch is added to the corresponding center, while the value associated with the saddle is assigned to the surrounding vertices. Thus, we obtain the same configuration for the child branch

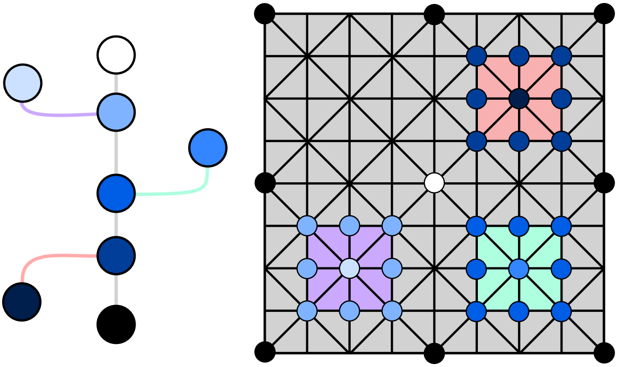


Figure 6.2: The topological landscape of the left contour tree can be constructed by refining the grid to add the child branches (red, green, and violet). The height of the nodes is encoded as shades of blue.

as we created for the root branch. This method allows for recursively constructing the landscape until all values are included.

Over the last years, extensions to the metaphor of topological landscapes have been proposed. The visualization of topological landscapes can be enriched by visualizing geometrical properties of the features [30]. Volume-preserving topological landscapes, as proposed by Harvey and Wang [139], can also be constructed for higher-dimensional scalar fields. Besides using topological landscapes for the analysis of scalar fields, this metaphor was also extended to other types of data like point clouds [250, 251, 252], general graphs [397] and dendrograms [207]. Demir et al. [76] proposed a hierarchical rendering for topological data of datasets with complex, deep hierarchies.

Contour trees have also been used to study uncertain data [196, 129, 383, 388], but none of the approaches allows for tracking changes over varying parameters or time. However, topological structures have been used for several years to track features over time. Edelsbrunner et al. [91] provide a theoretical algorithm to compute time-varying Reeb graphs. Such methods can be used for studying the temporal evolution of selected features [50, 366]. While Köpp and Weinkauf [194] use merge trees for temporally coherent layouts, they show a linearization of the volume data instead of the temporal evolution of the topological features. Bajaj et al. [22] propose a method to study single contour trees at different time steps, and they also discuss that time can be replaced by a parameter. This approach has been extended to higher dimensions and other topological properties [189]. However, they only allow interactively switching between timesteps without aiming for coherence. Sohn and Bajaj [319] use contour trees to visualize changes of isosurfaces over time but only work on selected isovalues. Another approach [330] dealing with contour changes over time works on cumulative effects of the split and merge events but does not take all events into account. Other approaches work purely on merge trees, for example, by comparing their subtrees [293] or computing their temporal evolution [252], but they do not include the additional information contained in split trees. None of these approaches visualize the temporal evolution of contour trees or the variation of contour trees over a parameter.

Closest to our work are the time-varying fuzzy contour trees [216, 215] which provide a contour tree alignment. This approach explicitly tracks the changes over time but visualizes the contour trees directly. Additionally, the authors do not propose a static overview visualization.

6.1.2 *Merge Tree Matching*

Existing approaches [216, 215, 252, 271, 194, 271, 272, 371, 372] also use mappings between either merge trees or contour trees.

Oesterling et al. [252] propose constructing time-varying merge trees by tracking the changes between two consecutive time steps based on linear interpolation. While their approach also allows for applying the procedure on split trees, their visualizations would need to be adapted to contain the entire topological information in the contour tree.

Also, other options for aligning contour trees exist and, thus, allow for tracking. A common way of computing distances between trees is the tree edit distance [35, 120] which can be minimized to obtain a mapping between two trees. This procedure has also been applied to merge trees [322, 293, 31]. The edit distance can be computed between two labeled trees by measuring the minimum number of operations for transforming the trees from one to another. Possible operations are inserting, deleting, or relabeling nodes, and each operation gets assigned a cost where the overall cost value is minimized. For computing the distance, it is necessary to specify the cost function where a cost function of merge trees can include the specifics of topological features [322].

The tree alignment distance, as used for contour tree matching by Lohfink et al. [215], is a variant of the tree edit distance that can be applied to unrooted trees like contour trees. The basic idea involves creating a super-tree that contains all trees that should be aligned as a sub-tree. This idea relates to our meta tree, which also contains all individual trees. However, as the matching algorithms differ, the structures of the meta tree and the super-tree also differ. For example, in contrast to Lohfink et

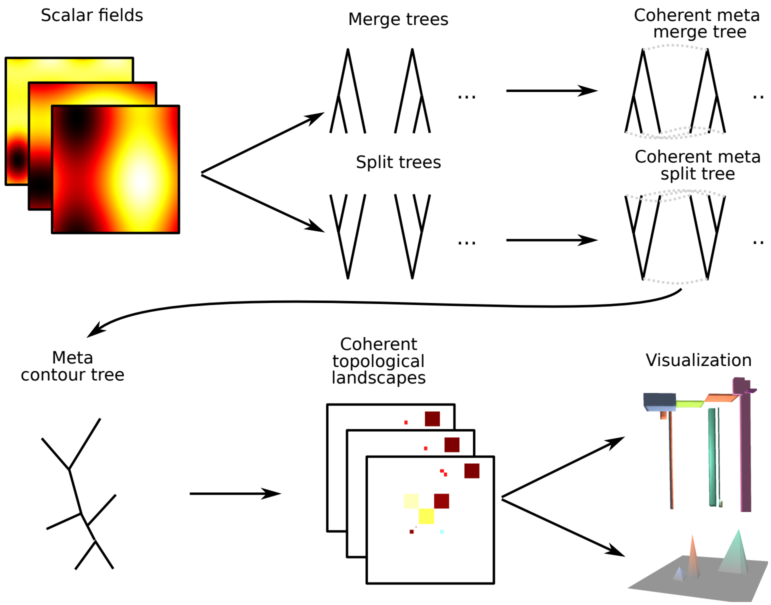


Figure 6.3: Our approach for computing and visualizing coherent topological landscapes is based on matching split and merge trees.

al., we allow matchings to violate the tree property which is stored in the meta tree as duplicated nodes with links (see Section 6.3.2 for more details).

6.2 OVERVIEW

Our approach for visualizing coherent topological landscapes consists of several steps shown in Figure 6.3. In the remainder of this chapter, we will assume that just one parameter is varied. A generalization to multi-dimensional parameter spaces, as we considered in the previous chapters, is briefly discussed in Section 6.6 as future work.

At first, the merge and split tree are computed for each scalar field individually, leading to one merge and one split tree for each discrete sample of the parameter range. For computing the merge and split trees, the implementation provided by TTK is used. TTK can also be used to simplify the trees, which is optional for our approach. However, especially for real-world data, this step is recommended to prevent the trees from becoming complicated by noise in the data. The merge and split trees are matched iteratively using the distance metric presented in Section 6.3.1. The trees are stored in a meta data structure where we obtain a meta split tree T_s and a meta merge tree T_m that store the split and merge trees for all parameter values (see Section 6.3.2 for more details). The meta trees can then be used to compute a parameter-varying contour tree.

This data structure is used to compute coherent topological landscapes based on coherent contour trees. Finally, we propose two options to visualize the variation of the topological landscapes over the parameter visualization as discussed in detail in Section 6.4. An animation of the topological landscapes is especially intuitive to show changes over time but imposes a high cognitive load. Therefore, we encode the topological landscapes as 2D scalar fields, which we stack ordered by the parameter value. The resulting 3D volume can be visualized using any volume visualization technique where we use direct volume rendering.

6.3 COHERENT CONTOUR TREES

For computing coherent contour trees, we match the merge and split trees as they are trees with a dedicated root node. This facilitates the matching in comparison to directly matching contour trees. To compute a meaningful matching, we assume a relatively high similarity between the trees that should be matched. Otherwise, a meaningful computation of matchings is not possible. However, as we aim at studying the parameter dependency of ensembles where we assume numerical parameters or, as an alternative, the temporal evolution of ensembles, we can assume that variations from one parameter setting to the next (or from one timestep to the following timestep) are sufficiently small.

6.3.1 *Distance Metric*

For matching the split and merge trees, a distance measure between the nodes of two trees is defined. For the distance measure, we differentiate between the leaves of the tree and the inner nodes because they represent different topological structures. As the root nodes of both trees represent the whole domain, they match by definition.

6.3.1.1 *Leaf nodes*

Leaf nodes correspond to isolated critical points because they represent either minima (for merge trees) or maxima (for split trees). A critical point i in the scalar field is defined by its spatial position x_i and its function value $f(x_i, p_i)$ where p_i denotes the parameter value which can also be replaced by the time. These two values should also be considered for the distance computation. Therefore, we define the distance between two leaves $\delta_L(i,j)$ as the weighted average of the distances $\delta_{L_1}(i,j)$, which considers the spatial distance between the points and $\delta_{L_2}(i,j)$ which considers the difference in the function values. The distances are defined as

$$\delta_{L_1}(i,j) = \left\| x_i - x_j \right\|$$

and

$$\delta_{L_2}(i,j) = |f(x_i, p_i) - f(x_j, p_j)|\,.$$

Before computing the weighted sum, both terms are normalized. The user can adapt the weights to define the influence of the two components. As a default, we weight $\delta_{L_1}(i,j)$ and $\delta_{L_2}(i,j)$ equally.

6.3.1.2 *Inner nodes*

For matching inner nodes, we take into account the corresponding spatial region, the position of the corresponding critical points, and the matches of the subtrees. Each merge or split node is surrounded by a region R_i that forms a connected component. In the case of merge trees, this region is formed by all spatial samples x_i for which $f(x_i) \leqslant f(x_j)$ where x_j is the spatial sample belonging to the parent node. For split trees, the samples that fulfill $f(x_i) \geqslant f(x_j)$ belong to the region R_i. For computing the distance between the regions of nodes i and j, we define a one-sided distance as

$$\delta'(i,j) = \frac{\sum_{x \in R_i \setminus R_j} \min_{q \in R_j} \|x - q\|}{|R_i|},$$

where $|R_i|$ describes the cardinality of the set of sample points. This computation can be accelerated by considering only the margins of the connected components. A two-sided distance can then be defined by weighting the one-sided distances with the region's cardinality leading to

$$\delta_{I_1}(i,j) = \frac{\delta'(i,j) \cdot |R_i| + \delta'(j,i) \cdot |R_j|}{|R_i \cup R_j|}\,.$$

For the similarity of the matching between the leaves of subtrees, we consider two subtrees S_i with root node i and S_j with root node j. The distance is computed as the average distance of all leaf pairs and can be written as

$$\delta_{I_2}(i,j) = \frac{\sum_{s_i \in S_i^L, s_j \in S_j^L} \delta_L(s_i, s_j)}{|S_i| \cdot |S_j|},$$

where S_i^L and S_j^L are the leaves of the subtrees S_i and S_j, respectively. Assuming that the nodes i and j match, we can compute the percentage of matches $\delta_{I_3}(i,j)$ of the leaf nodes of S_i and S_j. Finally, we consider the distance between the spatial positions $\delta_{I_4} = \left\|x_i - x_j\right\|$ which is defined analogeous to leaf nodes. Using $\delta_{I_1}(i,j)$ to $\delta_{I_4}(i,j)$, we define the distance for the inner nodes as the weighted sum.

As a default, we use equal weights for all components of the weighted sums.

6.3.2 *Matching*

For tracking the topology changes over a parameter, we create a data structure that stores the evolution of a merge tree or a split tree and refer to it as the meta tree. Note that the meta tree is not necessarily a tree, but the tree property can be restored, as discussed later. To cover the entire set of changes, we compute two meta trees, T_s and T_m, for the split and merge trees, respectively. These data structures can then be combined into a contour tree. The meta trees store the function values of the critical points, and their lifespans to enable precise tracking over time or parameter value.

We will first describe how to create a meta tree T from two merge or split trees T_1 and T_2. Additional trees can then be inserted iteratively. Generally, two nodes $a_1 \in T_1$ and $a_2 \in T_2$ are combined into a single node $a \in T$ if matched because the corresponding feature exists in both trees. This will also be reflected in the lifespan of this feature. If a node $b_1 \in T_1$ or $b_2 \in T_2$ has no matching node in the other tree, it corresponds to an emerging or disappearing feature. In this case, the nodes should also be included in the meta tree T, but the lifespan reflects the emergence or disappearance.

For creating the meta trees, we make use of the fact that merge and split trees are rooted trees, and the roots match by definition. Therefore, we start traversing the trees by matching the root nodes of T_1 and T_2. Then, we match the trees by applying a greedy algorithm when proceeding through the depth levels in a top-down procedure. For identifying matching nodes, we use the distance metrics presented in Section 6.3.1. However, in addition to searching the current depth level for a node in T_1 that matches a node in T_2, we also search the subsequent depth level.

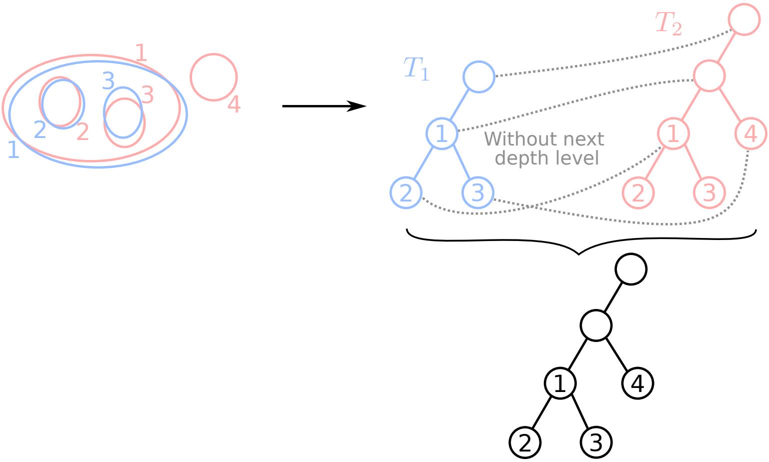


Figure 6.4: Isocontour 4, that emerges for the red level set, requires to match the trees T_1 and T_2 over the next depth level. The gray dashed lines indicate the matching without considering the different depth levels.

This procedure is also applied to search in T_1 for matches of the nodes in T_2. In this work, we only consider a difference of one depth level to reduce the computational cost because we assume the contour trees to be sufficiently similar. In Figure 6.4, we show an example that illustrates the necessity of this step. Here, a new isocontour with the label 4 emerges. This adds an additional depth level in T_2 compared to T_1. Without allowing for different depth levels, one might obtain a matching shown by the gray dashed lines. However, when allowing for a shift of one depth level, one obtains a meta tree as visualized in black.

Additionally, the hierarchical order of the critical points might change, as depicted in the example in Figure 6.5. Here, isocontour 1 is the parent of the isocontours 2 and 3 in the blue tree, but the parent of isocontours 3 and 4 in the red tree. Thus, to accurately match the trees, the node corresponding to isocontour 3 needs two parents in the meta tree T. Even though such a construction violates the tree property of T, we allow for it and describe how to restore the tree property for the contour tree creation later.

As we aim to investigate the variation of the topological structure over a driving parameter, this parameter induces an ordering of the different trees. Thus, we can iteratively add all trees following the order induced by the parameter p_i. Thus, the next tree T_3 is matched and added to

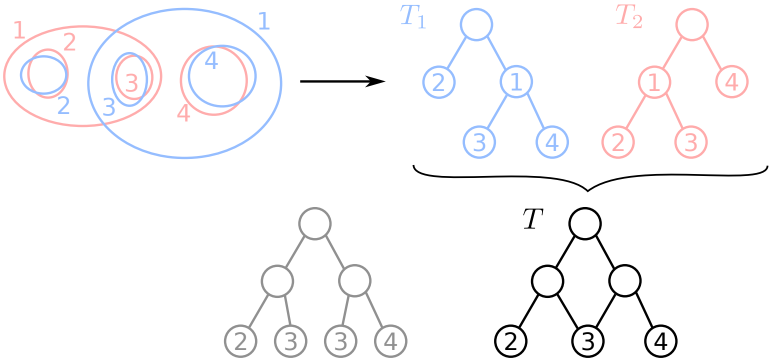


Figure 6.5: Changing the hierarchical order requires storing two parents to allow for matching the nodes corresponding to isocontour 3.

the meta tree T. To ensure the coherence, assume that T_2 is closer to T_3 than T_1 is. Therefore, we neglect the edges of T_1 that destroy the tree property and do not belong to T_2. In the end, the meta tree T will contain all the nodes and edges of all steps and, thus, the complete topological information.

This meta tree creation algorithm can be applied to the merge tree leading to a meta tree T_m as well as to the split tree for creating a meta tree T_s. The tree structure needs to be recreated to combine these data structures into coherent contour trees by following the algorithm proposed by Carr et al. [56]. The tree property is restored by duplicating the tree subtrees with more than one parent. To still keep the information that these subtrees are duplicates, they are linked to each other. Following the construction of the meta tree, at most, one of the subtrees is relevant for a given parameter setting (or timestep). Having restored the tree property, the contour tree can be created from the merge and split trees by directly applying the standard algorithm. For the coherent contour tree, all nodes and edges of all individual trees are stored. Additionally, we store the life spans of the features and the links between duplicates.

6.4 COHERENT VISUALIZATION OF TOPOLOGICAL LANDSCAPES

The meta contour tree described in Section 6.3.2 contains contour trees for each ensemble member. By extracting this contour tree, the topological landscape can be computed using the original algorithm presented

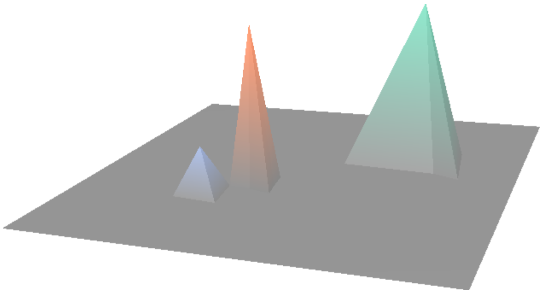


Figure 6.6: We directly render the topological landscape and create an animation over the parameter values. The different peaks are color-coded.

by Weber et al. [367]. As the landscapes are created in the same order, based on the coherent contour trees, the order of constructing the different topological features in the landscapes does not change. Therefore, the coherent contour trees directly lead to coherency in the computation of topological landscapes.

6.4.1 *Animation*

We propose two approaches for visualizing the topological landscapes in dependency on the parameter values. One possibility is using an animation to show the variation over the input parameter. This is especially intuitive if the variation over time instead of over parameter values is studied, as in this case, time is directly represented as time.

For creating the animation, each parameter value is mapped to corresponding time steps. For each parameter value, the topological landscape is visualized as a height field which can be directly constructed from the 2D scalar field. The visualization of such a time step is shown in Figure 6.6. We apply linear interpolation between the two height fields for smooth transitions between the topological landscapes for different parameter values. This also allows for smooth animations in case of varying step sizes between the parameter values in irregularly sampled parameter spaces. To visualize matching features, we apply a consistent color coding where each mountain in the landscape gets assigned a color.

However, switching between the duplicates introduced to provide temporal coherency might induce discontinuities in the visualization. These discontinuities represent changes in the topology encoded as switches

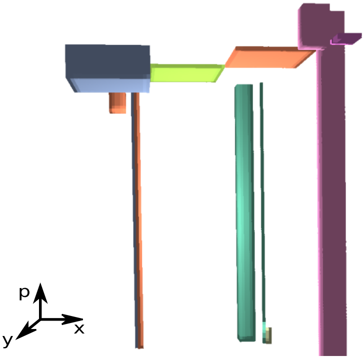


Figure 6.7: The topological landscapes can be converted to scalar fields, which can be stacked in order of the driving parameter. The resulting volume can be visualized as a direct volume rendering.

between duplicates. Therefore, they encode features of interest that should be visually pertained.

6.4.2 *Static Visualization*

While animations provide an intuitive visualization of the evolution, they might impose a high cognitive load if the users want to obtain a more global overview. Therefore, we propose a static visualization as an alternative. For the static visualization, each topological landscape is transformed into a 2D texture that contains the nodes' IDs. We explicitly encode the nodes' IDs instead of using the 2D height field to reduce the variation and avoid missing smaller features.

The resulting textures can be stacked on each other, ordered by the parameter value. Again, linear interpolation provides smooth transitions and accounts for different step sizes. Thus, we obtain a 3D volume representing the variation of the topological structure over the change of the parameter.

This volume can be visualized using standard volume visualization techniques like direct volume rendering, as shown in Figure 6.7. Designing a suitable transfer function allows for assigning unique colors to all nodes. For the remainder of this chapter, we use a categorical transfer function as also proposed by Weber et al. [367]. The transfer function can be adapted interactively to highlight special features or , for example, accentuate peaks whose height lies in a particular range.

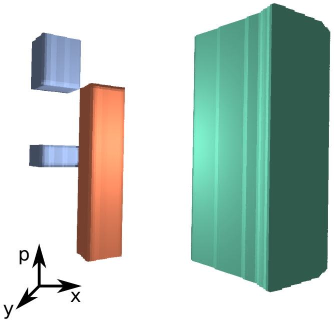


Figure 6.8: The stacked landscapes for the synthetic dataset are shown for the variation over driving parameter p. The green peak persists while the blue peak emerges, vanishes, and then replaces the orange peak.

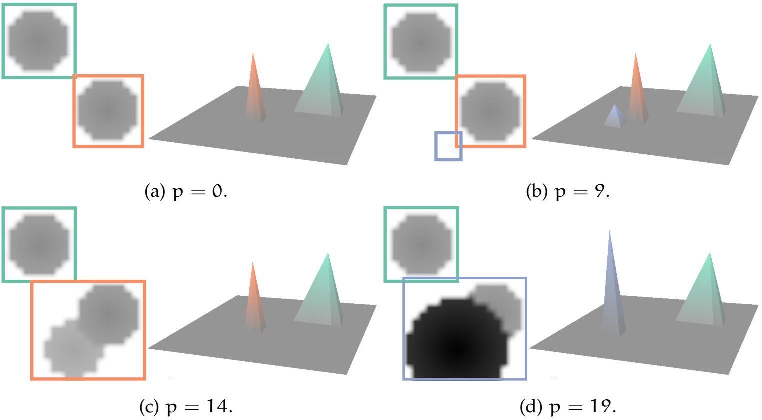
6.5 RESULTS

In the following, we will present applications of coherent topological landscapes for analyzing different datasets. After verifying our approach with synthetic data, we apply it to a 2D reaction-diffusion dataset. Finally, we apply the approach to a 3D simulation ensemble to study cavity flow.

6.5.1 *Synthetic Datasets*

We create a synthetic parameter-dependent 2D dataset to validate our approach. The dataset depends on a parameter p and contains 20 runs with a spatial resolution of 30×30. For each parameter setting, the dataset contains 2 peaks. A third one emerges with an increase of p and overlaps with one of the other peaks.

When computing the coherent topological landscapes, we simplify the contour trees by only keeping features with a persistence larger than 6% of the function value range. This reduces noise caused by creating the Morse function. The results for the synthetic dataset are shown in Figure 6.8. Observing the volume rendering of the stacked topological landscapes over the parameter variation provides an overview of the topological changes associated with parameter changes. The green peak persists for all parameter values. As the orange peak is also present for small values of p, it becomes clear that it corresponds to the other


(a) $p = 0$.

(b) $p = 9$.

(c) $p = 14$.

(d) $p = 19$.

Figure 6.9: When observing the animation of the topological landscapes, the evolution over the parameter can be observed. For each of the shown parameter values, the original data is shown on the left, where the peaks are surrounded by colored boxes whose colors match those used in the topological landscapes.

peak present from the beginning. The blue peak emerges over time as a separate peak and vanishes again. At some point, the orange peak vanishes, and at the same time, the blue peak emerges again.

To better understand this behavior, we investigate the animated rendering of the topological timesteps and compare it with the original data. The landscape and the underlying data for $p = 0$ are shown in Figure 6.9a, where the features in the data are linked to the features in the topological landscape by colored boxes. As expected, the two peaks in the data are visible as two peaks in the landscape. When increasing p, the third peak (color-coded in blue) emerges as shown for $p = 9$ in Figure 6.9b. When further increasing p, the blue peak merges with the orange peak as presented in Figure 6.9c, causing the blue mountain to vanish. Finally, for very large parameter values, the third peak dominates. This leads to the switch in topological features, encoded in the rendering of the stacked landscapes as a vanishing of the orange feature and the emergence of the blue feature.

Note that a switch in the topology is visually encoded as a sudden switch in the landscapes. Thus, the topological change is clearly visible.

6.5.2 *Pattern Formation in 2D*

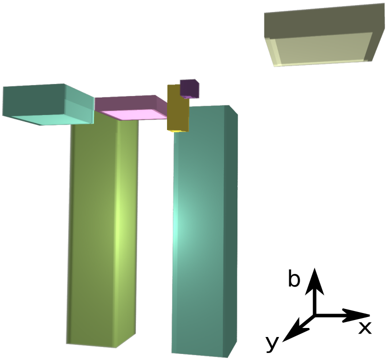
Next, we apply our approach to studying self-organization in pattern formation. One group of systems that shows pattern formation is the so-called reaction-diffusion systems. While this mathematical model can be applied in various domains, for example, in biology, the most prominent examples are applications in chemistry. Chemical reactions transform the different components in the equation into each other, while diffusion allows for spatial spreading. This chapter investigates a reaction-diffusion system described by a two-layer brusselator model [390]. For example, this kind of model could be used to describe the behavior of two thin layers of gel that meet at an interface. The equations used for the simulation are provided in Appendix B.2. While the equations depend on 4 parameters, we only consider the variation of parameter b between 7.5 and 9.0 using adaptive steps between 0.01 and 0.1. This leads to 25 time-varying runs with a resolution of 32×32.

As we aim to investigate the relationship between the emerging patterns and the driving parameter b, we only use the last time step, where the pattern is fully formed. For the computation, we only consider features with a persistence larger than 15% of the whole value range to exclude noise and focus on the most relevant features.

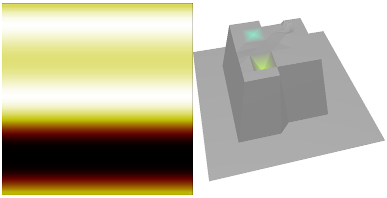
The results for this dataset are shown in Figure 6.10. The stacking of the topological landscapes shows little variation over a large range of parameter values. However, for larger values of b, additional features appear. When comparing the topological landscapes with the original data as shown in Figures 6.10b and 6.10c, one observes a change in the pattern. The increasing number of features already pointed towards a topologically more complex pattern which agrees with the change from a stripe pattern to a spot pattern. Thus, our approach does not only allow us to quickly identify the parameter value at which the topological structure changes but also the topological complexity of the simulation output in different parameter ranges.

6.5.3 *Cavity Flow in 3D*

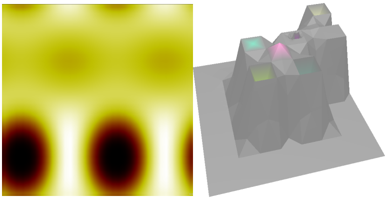
Finally, we apply our algorithm to a 3D simulation ensemble studying lid-driven cavity flow. The lid-driven cavity flow, whose creation is explained in more detail in Appendix B.3, describes the flow in a cavity that is induced by moving the top lid. Each of the 10 simulation runs consists of 99 time steps with a spatial resolution of $32 \times 32 \times 32$. We consider the magnitude of the flow velocity field that depends on the Reynolds number Re. For this dataset, we investigate the dependency on



(a) Stacked landscapes.



(b) $b = 8.5$.



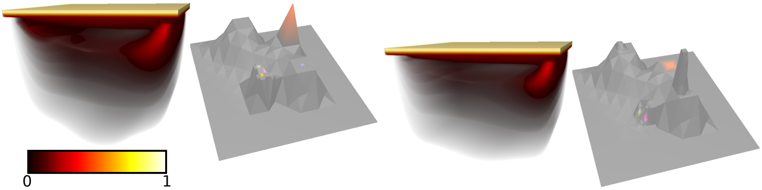
(c) $b = 9.0$.

Figure 6.10: The stacked landscapes for the reaction-diffusion dataset show a change in the topology for large parameter values b. The different topological features can be investigated when comparing the landscapes of the different timesteps to the visualization of the data (see b and c). Here we see that the pattern transitions from stripes to a spot pattern.

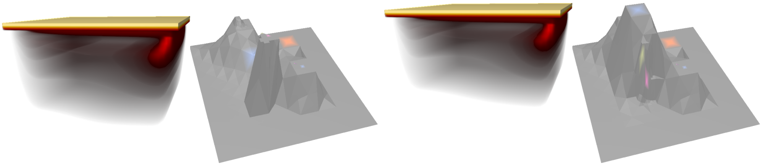
the parameter value as well as the temporal evolution. For this dataset, we remove noise by keeping only features with a persistence larger than 2% of the data range.

We start by analyzing the influence of the parameter. Therefore, we choose the last time step, where the flow pattern is fully formed. The results for specific values of Re are shown in Figure 6.11. With an increase in the Reynolds number, we observe a variation in the number of peaks in the topological landscapes indicating variations in the complex dataset topology. We also observe features occurring in each simulation run, like the orange, relatively central peak. However, this peak decreases in height with an increasing Reynolds number. The topological changes might be explained by the flow becoming less laminar with an increase in the Reynolds number.

To study how the flow patterns emerge, we investigate the temporal evolution of the data. Here, we choose $\text{Re}= 1000$, which corresponds to a more complex landscape. When viewing the animation, one can identify features that emerge early and persist over the simulated time. Figure 6.12a shows the topological landscape and a volume rendering of the scalar field after ten timesteps. Several features emerge and vanish



(a) Re= 200. (b) Re= 400.



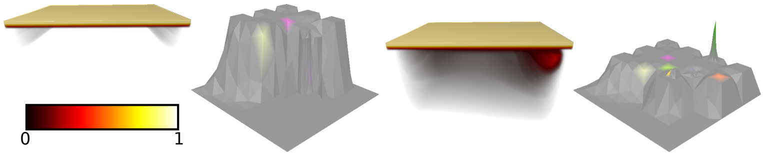
(c) Re= 600. (d) Re= 800.

Figure 6.11: Observing the coherent landscapes for different values of the Reynolds number Re reveals that the number of topological features increases if the Reynolds number increases. Here, the volume renderings of the scalar fields (left) are shown together with the topological landscapes (right)

over time until we reach the final state shown in Figure 6.12b. When comparing this topological landscape to the one at the tenth timestep, we observe that the peaks persist, but also additional peaks are present. Thus, our approach allows for tracking the evolution of the topological structure over time as well as over a parameter value. This application shows that our approach can also be used to study topological features in 3D data whose dimensionality is reduced to 2D by creating the topological landscapes.

6.6 DISCUSSION

In this chapter, we presented an approach for investigating topological changes over the variation of an input parameter of the ensemble or over time. We create coherent topological landscapes by matching the nodes in merge and split trees to create coherency between contour trees. The topological landscapes can be visualized by rendering them as an animation over the parameter value, but we also include a static visualization by stacking the landscapes ordered by the parameter value. The resulting volume can be shown using direct volume rendering. We verified our approach on a 2D dataset and also applied it to 2D and 3D simulation ensembles.



(a) After 10 time steps.

(b) End of simulation.

Figure 6.12: In the temporal variation of the topological landscapes for the lid-driven cavity flow, the first features emerge, as can be seen in the volume rendering as well as in the topological landscape (a). The flow velocity magnitude field increases in complexity leading to a more complex topological landscape (b). The features that could already be observed at timestep 10 persist until the end of the simulation.

Our visualizations show abrupt changes if the underlying topology changes (see Section 6.5.1). While this can be desirable because the sudden changes in the visualization encode changes in the topology, the scalar field does not change suddenly. Therefore, the sudden changes might be undesirable, especially for analysis goals that do not require highlighting the changes in topology. Future work could consider avoiding these sudden changes in the visualization.

Another aspect that can be improved is the computational efficiency. Current computation times are in the range of seconds up to a few minutes, even though the datasets are comparatively small. However, as the creation of the meta tree structure forms a pre-processing step, an interactive analysis can still be achieved with this approach. The computation times could be sped up significantly, for example, by parallelizing the approach.

For matching the nodes representing topological features, we proposed to minimize a distance metric that covers different facets, can be generally applied, and flexibly adapted as the users can modify the weights. On the other hand, this includes the weights as a set of parameters for our approach. While a suitable choice of parameters is important for meaningful results, we found that no extensive parameter tweaking was necessary for the use cases presented in the paper. However, the impact of the different facets requires a better understanding. Also, a more detailed comparison to other matching approaches, as presented in Section 6.1.2, would be beneficial. Our visualizations of the variations over the parameter values only require the discussed meta tree structure, which can also be created by using other merge tree matching approaches. Therefore, a detailed study that compares the different approaches and provides a guideline on when to use which matching strategy would be beneficial.

Finally, the current implementation of the approach only supports investigating the variation over one parameter. However, as discussed in the previous chapters, simulation ensembles often depend on multiple parameters. Therefore, the approach could be extended to cover the variability over multiple parameters. One could include the time component, leading to a flexible approach for studying the topological variation for multiple facets of the ensemble data. While the iterative matching could be generalized to multiple dimensions, the order of the matching would significantly influence the results, which requires finding a suitable starting point and order for matching the additional trees.

7

SPATIAL GLOBAL SENSITIVITY ANALYSIS

After aiming to understand the qualitative influence of the parameters on the simulation outcome, it is also important to obtain an overview of which parameter influences the simulation result most and how these values vary over the spatial domain. In global sensitivity analysis, typically, one or more sensitivity values per parameter and simulation output are computed. In simulation ensembles it is desirable to understand the spatial variations in the sensitivity to input parameters. For example, this is important in medical simulations. Close to risk structures like tumor tissue, less sensitivity is desired for successful treatment without harming too much healthy tissue, while in other regions, a higher sensitivity might be tolerable.

If the spatial sensitivity values are computed, one obtains one scalar field per input parameter. Therefore, it is necessary to analyze multiple fields to cover the sensitivities of all parameters. Visually analyzing 3D multi-field data is challenging, especially due to occlusion problems. This is one of the reasons that spatial sensitivity analysis is still little explored [295].

In this chapter, we propose a visual analysis approach for interactively investigating the spatial distribution of parameter dependencies. Our approach builds around a novel overview visualization that supports a global overview of all spatial locations without suffering from occlusion like 3D volume visualizations. For this purpose, we apply data-driven space-filling curves to multi-field data and analyze the properties of our adaptations. We embed this overview visualization in a visual analytics solution that also allows for analyzing the sensitivities on different levels of detail and visualizing the qualitative dependency of the simulation output on the input parameters. While our approach can be applied to different sensitivity computation methods, we discuss three methods and their advantages and drawbacks. By comparing the algorithms used in our approach to alternative choices, we derive clear guidelines for choosing the best algorithm for the corresponding application. Finally, we show the utility of our approach by presenting the analysis of two real-world use cases from the medical domain.

We start with providing the relevant background for the techniques used in this chapter in Section 7.1. After specifying the problem in Section 7.2, we provide an overview of our solution in Section 7.3. Our design choices are described in Section 7.4. After an algorithmic evaluation (see Section 7.5) and showing use cases (see Section 7.6), we discuss the limitations of the presented work (see Section 7.7).

The results in this chapter are based on the following manuscript:

> **M. Evers**, S. Leistikow, H. Rave, and L. Linsen, Interactive Visual Analysis of Spatial Sensitivities, *to be submitted*

Simon Leistikow created the aneurysm simulations and provided feedback on the analysis. Hennes Rave implemented data-driven space-filling curves in Voreen. I implemented all other parts of the approach except the space-filling curves and created the results. All authors contributed to writing and editing the manuscript of the paper.

7.1 RELATED WORK AND BACKGROUND

While sensitivity analysis was briefly mentioned in the context of parameter space analysis (see Section 2.2.1), we provide a more detailed view of the current state of the art before presenting the background of the sensitivity computation and visualization used in this chapter.

Several comprehensive overviews about sensitivity analysis are available [264, 136]. Approaches in the field of sensitivity analysis are commonly divided into local and global approaches [298]. Local sensitivity analysis methods investigate how small changes in a parameter influence the simulation outcome while the other parameters remain fixed. These methods have also been included in visual analysis approaches [34, 266, 33, 49]. However, while local approaches are often computationally more efficient, they only provide a local perspective and might miss important features, especially for large parameter spaces. Global sensitivity approaches, instead, consider the whole parameter space but are only tackled by few visual methods. Sobol indices, which are among the most popular global sensitivity estimators (see Section 7.1.1.1), can be visualized by Fanovagraph [116] using a graph-based visualization which was extended by Yang et al. [389]. Ballester-Rippol et al. [23, 25, 24] propose using tensor-train models to compute Sobol indices efficiently. However, they focus on the sensitivity of single scalar outputs and do not include the spatial component. Applying their methods to spatial data would be computationally very expensive because each sample requires a tensor train surrogate. An alternative to the quantitative investigation of parameter space analysis are more qualitative approaches [221, 225]. Surrogate models like InSituNet [148] can also be used to derive the sensitivities to the input parameters directly. Note that we consider the sensitivity to input parameters by using ensemble data. Ensemble sensitivity analysis (ESA), however, usually refers to a technique for studying the sensitivity on the initial conditions [341, 197].

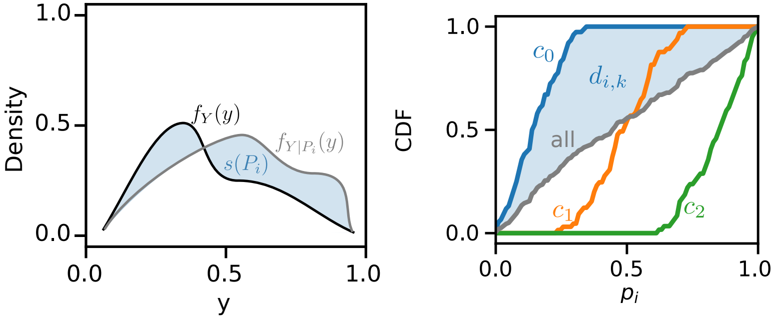
These discussed approaches do not consider the spatial variation of the sensitivities even though spatial sensitivity analysis became more popular over the last years [209, 387]. Due to the lack of better encodings, the spatial variations are usually shown side-by-side. However, Şalap-Ayça et al. [295] experiment with stacked views. Their visualization is only applicable to 2D data, and the authors identify the visualization of spatial sensitivities as an open research question. Closest to our work is the approach by Biswas et al. [37], who investigate spatial variations in the sensitivity of temporal data. They use maps combined with clustering, so their visualizations are also limited to 2D data. Thus, to our knowledge, our approach is the first one that allows for visual analysis of spatial sensitivities for 3D domains.

7.1.1 *Global Sensitivity Measures*

In the following, we will present a brief overview of the basic ideas and potential limitations of different sensitivity analysis methods used in this thesis. Note that the sensitivity measure for our approach can be easily exchanged depending on the data and analysis tasks that should be analyzed. Other popular sensitivity measures that we did not consider include the Fourier amplitude sensitivity test (FAST) [299] and the Morris method [234].

7.1.1.1 *Sobol Sensitivity Indices*

Sobol indices [318, 298] are one of the most common global sensitivity analysis methods. The Sobol decomposition, which belongs to the analysis of variance (ANOVA) methods, is based on distributing the total variance $D = \mathrm{Var}[f]$ of a function f to the variances of the individual parameters and their combinations. It assumes that a function can be written as a sum of subfunctions where each of these subfunctions depends on a subset of the input parameters and, thus, assumes that the input parameters are independent. Using this decomposition, it is possible to decompose the total variance as $D = \sum_{\alpha} V_{\alpha}$ where α is a tuple that refers to the included variables. The Sobol indices are defined as



(a) δ.

(b) DGSA.

Figure 7.1: The basic idea for computing the similarity measures δ, and DGSA is considering the change in distributions when fixing initial parameters. a) δ uses the density function of the output. b) DGSA considers the shift in the CDFs between different clusters c_i.

$S_\alpha = V_\alpha / D$ leading to $\sum_\alpha S_\alpha = 1$. Thus, the Sobol indices indicate which fraction of the variance corresponds to each parameter (or combination of parameters).

In addition to the high number of indices S_α, there are derived indices. Among the most common ones are the total indices which attribute the complete effect to a set of input parameters α. They are defined as the sum of all Sobol indices whose tuples are not disjoint with α, which can be written as $S_\alpha^T = \sum_{\{\beta | \beta \cap \alpha \neq \emptyset\}} S_\beta$.

Commonly, Sobol indices are computed by using Monte Carlo methods. As many samples are needed, and the number of samples scales poorly with the dimensionality of the parameter space, often surrogate models are used. Surrogate models approximate the simulation in a computationally efficient way, thus allowing for obtaining a higher sampling of the parameter space. However, finding a suitable surrogate model is a challenging and non-trivial task. A stable yet efficient computation of Sobol indices can be achieved using dedicated sampling strategies that create uniform samples of the parameter space. The sequence initially proposed by Sobol [318] has been extended by Saltelli [297] for more efficient computation. When computing the sensitivity indices in this work, we used the implementation provided by SALib [156, 164].

7.1.1.2 *Delta Sensitivity Measure*

An alternative metric to compute global sensitivities is a measure called δ [43]. While this measure is popular in operations research, it has also been used by Biswas et al. [37] to visually investigate the sensitivity in weather ensembles. This sensitivity measure is built upon investigating variations in the simulation output's density function for variations in the input parameters as shown schematically in Figure 7.1a.

Let $f_Y(y)$ be the density function for output value y where the total output Y with a free variation of all parameters is considered. $f_{Y|P_i}(y)$ denotes the density if parameter P_i is fixed to value p_i. For computing the sensitivity measure δ_i, the shift $s(P_i)$ between these two density functions is considered, leading to

$$s(P_i) = \int |f_Y(y) - f_{Y|P_i}(y)| dy\,.$$

The sensitivity measure δ_i can then be defined as

$$\delta_i = \frac{1}{2} E_{P_i}\left[s(P_i)\right]\,, \tag{7.1}$$

where $E_{P_i}\left[s(P_i)\right]$ denotes the expectation value of $s(P_i)$ which can be computed as

$$E_{P_i}\left[s(P_i)\right] = \int f_{P_i}(p_i) \left[\int |f_Y(y) - f_{Y|P_i}(y)| dy\right] dp_i\,.$$

Thus, one obtains a value δ_i with $0 \leqslant \delta_i \leqslant 1$ [43]. Stronger expected shifts in the density function f lead to higher values of δ_i and, thus, indicate a stronger sensitivity on the parameter P_i. As discussed by Borgonovo [43], this sensitivity measure is global, quantitative and model-free, which means that no prior assumptions about the model are necessary. Additionally, strong assumptions about the sampling scheme, like for the efficient computation of Sobol indices, are not required.

While Equation 7.1 only defines the computation of the sensitivity measure δ for single parameters, the extension to investigation interactions between multiple parameters is straightforward by fixing more than one parameter. For this work, we also used the implementation provided by SALib [156, 164] but removed the additional computation of Sobol indices to obtain comparable timings.

7.1.1.3 *Distance-Based Generalized Sensitivity Analysis*

A third alternative is the distance-based generalized sensitivity analysis (DGSA) proposed by Fenwick et al. [103]. DGSA is a global sensitivity analysis method based on clustering and cumulative distribution functions (CDF). Other than the efficient computation of Sobol indices but similar to the δ sensitivity, it does not depend on specific sampling functions and can be applied to existing datasets that were created for multiple purposes. Additionally, it can be applied to other than scalar data.

The main idea of this approach is to divide the simulation outcome into similar groups and calculate the distances between the CDFs for each of these groups to quantify their differences in parameter space. Therefore, we first need to find a clustering of the ensemble members. The approach initially proposed by Fenwick et al. [103] uses k-medoids clustering. It works similarly to k-means but uses actual data points as centers for the clusters. The parameter k denotes the number of clusters and depends on the dataset.

In this chapter, we work on scalar values. Therefore, we choose the deterministic Fisher's natural breaks algorithm [109] which copes well with multi-modal distributions of the data. Similar to k-means, the algorithm minimizes the average deviation from the mean of the cluster while maximizing the deviation from the means of the other clusters. Due to the 1D structure of the data, a dynamic programming approach can be used which finds a global optimum. However, similar to k-means or k-medoids, Fisher's natural breaks algorithm requires the number of clusters k as an input parameters.

We apply DGSA to spatially varying data and also the number of clusters might vary spatially. In general, the choice of a suitable value for k is a non-trivial task. Therefore, we include an automatic screening in which we compute the clusters for $k = 3$ to $k = 10$ and compute silhouette coefficient in each case. Using the silhouette coefficient to estimate the number of clusters is a standard method [393]. As the final value for k, we choose the clustering with the highest silhouette coefficient. To ensure a sufficiently large sample size in each cluster, clusters with less than ten members are not allowed.

In the next step of the algorithm presented by Fenwick et al. [103], a CDF $F(p_i|c_k)$ can be computed for each identified cluster c_k and for each parameter p_i. If the data is sensitive to parameter p_i, the functions differ, while similar functions indicate no sensitivity. Quantifying these differences, thus, provides a sensitivity measure for this parameter.

The distance between two CDFs can be computed by calculating the area between the functions as shown in Figure 7.1b. In the case of DGSA, we want to calculate the distance $d_{k,i}$ between a CDF $F(p_i|c_k)$ for a single cluster c_k to the CDF $F(p_i)$ of all samples in the whole ensemble:

$$d_{k,i} = \int_{p_{i,\min}}^{p_{i,\max}} |F(p_i|c_k) - F(p_i)| dp_i \,.$$

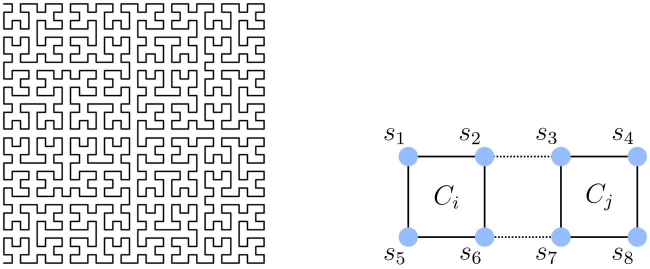
However, directly interpreting the distance values is difficult. To make the distances generally interpretable, the statistical significance of the distance is computed. A bootstrapping procedure is used to calculate the 99% quantile of distances in randomly selected subgroups. Therefore, we select B randomly sampled subsets of the original dataset. We calculate the distance to the total CDF $F(p_i)$ for each subset. After determining the 99% quantile $\hat{d}_{k,i}$, we can reject the null hypothesis that there is no difference between the different distributions for the clusters c_k if $d_{k,i} > \hat{d}_{k,i}$ for any k. To make the outcome easier to interpret and reduce the sensitivity computations to a single number, the computed distances can be normalized as $d^S_{k,i} = d_{k,i}/\hat{d}_{k,i}$. The simulation outcome can now be considered sensitive to parameter p_i if $d^S_{k,i} > 1$. Averaging over the normalized $d^S_{k,i}$ for all clusters c_k leads to a quantitative sensitivity value for parameter p_i. However, computing the average of the distances for K clusters as $s(p_i) = \frac{1}{K}\sum_{k=1}^{K} d^S_{k,i}$ reduces the influence of outliers. This might result in $s(p_i) < 0$ even if single clusters show a statistically significant deviation from the CDF of the complete ensemble. However, depending on the analysis goals, one might choose the maximum of the values $d^S_{k,i}$ instead of the average. In this chapter, we analyze the overall sensitivities and use the average.

For this chapter, we included an implementation of this approach into Voreen. The advantage of this sensitivity measure is easy interpretability: The normalization allows for a straightforward interpretation of whether the output is sensitive to a parameter. Therefore, it is advantageous if factors other than parameters influence the outcome, such as initial conditions. However, the output quality strongly depends on the choice of the parameters (see Appedix A.3), such as the number of iterations which are not necessary for the other methods presented in this chapter.

7.1.2 *Space-filling Curves*

A space-filling curve (SFC) is a curve that covers all points in a multidimensional space. Various SFCs have been proposed over the last few years. Among the most well-known ones are the Peano curve [256] and the Hilbert curve [157] (see Figure 7.2a). These SFCs are constructed recursively and preserve the locality well. However, they do not consider the underlying data, which might lead to splittings of features when plotting the data over the SFC. This was also observed by Zhou et al. [399], who address this problem by adapting context-based space-filling curves [74]. *Data-driven space-filling curves* [399] define an objective function considering local variation and data coherency.

In the first step, Zhou et al. [399] create a dual graph of the graph of small circuits. For the optimization function W, they use a weighted av-



(a) Hilbert curve.

(b) Circuits for data-driven space-filling curve.

Figure 7.2: Space-filling curves like the Hilbert curve (a) visit every spatial position. A data-driven space-filling curve can be computed by joining circuits C_i and C_j while minimizing an optimization function based on the data values s_i (b).

erage of a term for value coherency and a term for locality preservation. The cost $W(C_i, C_j)$ of adding circuit C_j is defined as

$$W(C_i, C_j) = (1-\alpha)N(C_i, C_j) + \alpha R(C_i, C_j)\,.$$

where C_i and C_j denote adjacent circuits, α is the weight factor, N the feature preservation term that is also used in context-based space-filling curves [74] and R the term covering locality preservation. Note that both terms are normalized to the range $[0, 1]$. The feature preservation term can be written as

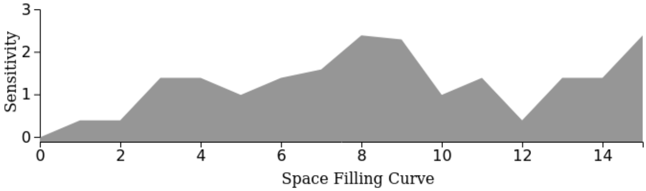
$$\begin{aligned} N(C_i, C_j) = d(s_2, s_3) + d(s_6, s_7) + d(s_3, s_4) + d(s_7, s_8) \\ + d(s_4, s_8) - d(s_2, s_6) - d(s_3, s_7)\,, \end{aligned} \tag{7.2}$$

where s_i are the scalar data values on the different grid positions as shown in Figure 7.2b and $d(s_a, s_b) = |s_b - s_a|$ denotes the distance function between two vertices. The spatial domain is divided into a user-defined set of blocks for local coherency. The locality preservation term is defined as

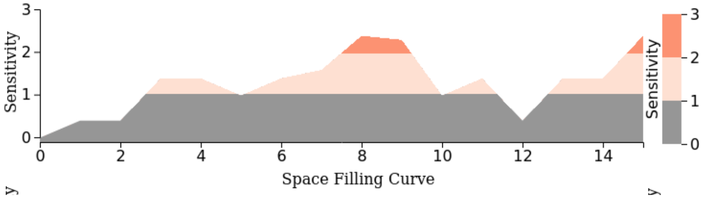
$$R(C_i, C_j) = \left\| (C_j.x, C_j.y) - (S_{C_j}.x, S_{C_j}.y) \right\|\,,$$

where S_{C_j} is the center of the block. As Zhou et al. [399] found that $\alpha = 0.1$ produces good results, we also use this factor in this chapter.

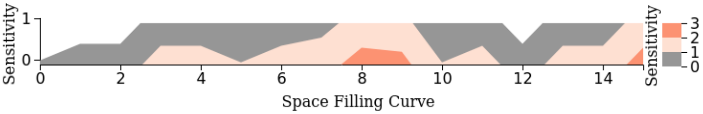
SFCs allow for linearizing the data and, thus, reducing the dimensionality to 1D. Recently, 1D projections of spatial data, including ensembles, have been used to detect patterns [113, 54, 384]. Ensemble data have also



(a) Area chart.



(b) Division in bands.



(c) Move bands to baseline.

Figure 7.3: Horizon charts are created by using an area chart (a), dividing it into bands (b), and moving them to the baseline (c).

been analyzed using enhanced line charts [77]. The analysis of volumetric ensemble data can also be supported by using a nonlinear scaling of a Hilbert curve [370]. However, all of these approaches focus on visualizing the simulation output and on ensemble variations. Thus, this approach cannot be applied to visualize the spatial sensitivities as these volumes might differ significantly.

7.1.3 *Horizon Graphs*

Horizon Graphs [107, 283] were initially proposed to visualize a large set of time series. They use small multiples to visualize each time series individually. Layered area charts reduce the space required for each time series while allowing for an accurate reading of the data values.

The first step in creating horizon graphs is the creation of an area chart as shown in Figure 7.3a. The graph is then horizontally divided into discrete, equally sized bands where the number of bands is fixed. Thus, the size of the bands is adapted to the data. The different bands are color-coded using a suitable color scheme to convey the bands' different heights, as seen in Figure 7.3b. In the last step, the height of the visualization is reduced by collapsing the different bands and showing them on top of each other, as shown in Figure 7.3c.

Horizon graphs have several advantages in the visualization of one-dimensional data. Besides allowing for reading off values precisely, they avoid visual clutter that often occurs if several line charts are shown together. It also reduces the need for space compared to showing individual line charts for each curve. On the other hand, horizon graphs need a lot of vertical space, which is not always available, especially when using multiple linked views. At the same time, the spatial separation of individual series makes comparisons less accurate compared to combined line charts.

7.2 PROBLEM SPECIFICATION

Similar to the previous chapters, we investigate a simulation ensemble with $|P|$ input parameters, which we assume to be numerical. The simulation run r_j is characterized by the parameters $P_i(r_j)$ with $i = 1, \ldots, |P|$. We only consider scalar simulation data for this work such that the simulation output is a single, two- or three-dimensional scalar field.

This chapter aims to analyze the simulation output's dependency on the input parameters. Therefore, we identified four individual tasks that have been discussed in the literature as important for ensemble analysis:

T7.1 Determining the *quantitative influence of individual parameters*. The simulation outcome's sensitivity on the input parameters should be quantified. This allows for identifying the most relevant parameters, which could be used for a denser sampling in this parameter. Identifying parameters of less influence allows for providing fewer samples in that parameter while creating simulations and allows for putting less focus on the dependency on this parameter during the analysis process.

T7.2 Analyzing *spatial parameter sensitivities*. Different spatial regions often vary in their importance for data analysis. For example, the regions around risk structures in medical applications are commonly more interesting than other spatial regions. The sensitivity measure is evaluated in each spatial position individually to obtain the spatially resolved sensitivities. These spatially resolved sensitivities should then be evaluated to identify regions of high or low sensitivity.

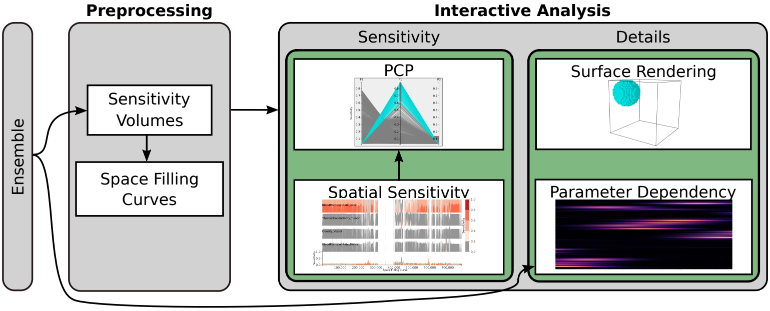


Figure 7.4: Sensitivity volumes are computed in a preprocessing step and can be interactively analyzed.

T7.3 Investigating *relations between the sensitivities to different parameters*. Instead of investigating the sensitivity of single parameters one after the other, the sensitivities should be considered together to identify influences between the different parameters. Studying pairwise correlations between the sensitivities to different parameters should also be possible to support a better understanding of the underlying model.

T7.4 Visualizing the *qualitative dependency of the simulation outcome on the input parameters*. It should be possible to observe how the simulation outcome varies with the input parameters in selected spatial regions of interest, which might be regions strongly sensitive to one or several input parameters (see task T7.1). For example, the users should be able to identify whether the simulation result increases or decreases in the respective region.

7.3 OVERVIEW

To address the different analytical tasks, we follow a visual analysis pipeline presented in Figure 7.4. We compute sensitivities for each spatial sample to study the quantitative dependency (task T7.1) and support a spatially resolved analysis (task T7.2). In this work, we include three different sensitivity measures, but they could be easily exchanged. We obtain one volume of sensitivity measures for each parameter by computing the spatially resolved sensitivities. Note that it is also possible to directly cover interactions between parameters in the sensitivity measure, for example, by considering higher-order Sobol indices. The resulting additional sensitivity volumes can be directly included in the analysis. Thus, we obtain at least |P| volumes, one for each parameter, which should be analyzed together. In the following steps of the analysis process, we focus on analyzing this multi-field data.

The multiple sensitivity values should be visualized to analyze the influence of the different parameters on the outcome (task T7.1). Thus, we want to analyze spatially resolved multi-dimensional data. This is a common multi-dimensional data visualization problem. Therefore, we choose parallel coordinates (PCP) to visualize all spatial samples. Our design choices are discussed in detail in Section 7.4.1. In the PCP, each axis corresponds to the sensitivity to one parameter, and each polyline corresponds to one spatial sample. PCP allows for analyzing the distribution of the sensitivity values while also studying the correlations between adjacent axes, which supports investigating relationships between parameters (task T7.3). Interactively reordering as well as brushing on the axes facilitates the analysis. Additionally, it scales relatively well with dimensionality (corresponding to the number of parameters).

However, PCPs do not encode spatial information. Therefore, we allow brushing so users can select sensitivity ranges on individual or multiple axes. The spatial samples in the selected ranges are visualized in the spatial domain. This allows for investigating the spatial distribution of the sensitivity values (task T7.2). A surface rendering shows the corresponding regions selected by brushing. Note that we want to visualize a binary volume of selected voxels. Therefore, we decide to show the surfaces of the voxels, which leads to discrete steps in the rendering. However, they are no undesired artifacts but instead prevent misinterpretations.

While these coordinated views provide spatial context to the parallel coordinates, this method does not provide an overview for obtaining a global understanding of the spatial sensitivity distributions (task T7.2). Showing all sensitivity values in all fields at once in a 3D visualization would also lead to large amounts of clutter. Therefore, we propose a new visualization for obtaining an overview of spatial sensitivities. The sensitivity values are visualized over a 1D linearization based on a precomputed space-filling curve (SFC) (see Section 7.4.2.1). We visually encode the data using a combination of Horizon Graphs and line charts. Thus, we obtain a scalable overview of the multi-field sensitivity volumes.

Investigating the sensitivity values indicates the quantitative sensitivity to the input parameters. Especially for parameters with high sensitivity, a more quantitative sensitivity analysis is of interest. Thus, the explicit dependency of the simulation output on the input parameter should be visualized (task T7.4). This task is addressed using a heatmap-based visual encoding which we refer to as parameter dependency visualization. It shows the simulated values of different spatial positions over a selected input parameter. We also use the SFC to linearize the spatial positions to obtain coherency with the other visualizations. More details are presented in Section 7.4.3. Thus, the users can identify patterns in dependency on the input parameters.

We use brushing and linking to coordinate the different views. The data visualized in the spatial visualization and the parameter dependency visualization is defined by brushing in the PCP. Brushing is not only possible in the PCP but also to select spatial regions in the spatial sensitivity visualization. The lines representing the spatial samples in the corresponding regions are highlighted in the PCP and shown in the spatial visualization. The spatial sensitivity visualization, by definition, already contains spatial information about the data. However, it is required to also show the spatial regions explicitly in a 3D encoding for providing the spatial context in 3D space. It also allows for differentiating between features of interest and possible artifacts caused by the linearization of the domain using the SFC.

We implement our approach as a web-based analysis tool using Dash, Plotly [269], D3 [45], and vtk [303].

7.4 VISUAL DESIGN

The volumes containing the sensitivities to the different parameters are formed by computing the sensitivity measure for each spatial sample separately. While some measures also allow for computing indices that measure the interactions between the parameters, like second-order Sobol indices, we do not treat them explicitly in this work. These interactions are not conceptually different from the single parameter sensitivities and, thus, can be directly included in the analysis approach.

7.4.1 *Parallel Coordinates Plot*

We use parallel coordinates to obtain an overview of the quantitative sensitivities (task T9.1). We want to analyze the parameter values for each parameter for all spatial samples, corresponding to a multi-dimensional data visualization task. PCP is a common approach for visualizing scalar multi-field data as it scales well with the number of dimensions and contains the complete information of all fields without the need to aggregate. Parallel coordinates scale better than table-based approaches or SPLOMs and, thus, allow for analyzing higher-dimensional parameter spaces or including interactions between parameters, which would significantly increase the number of fields.

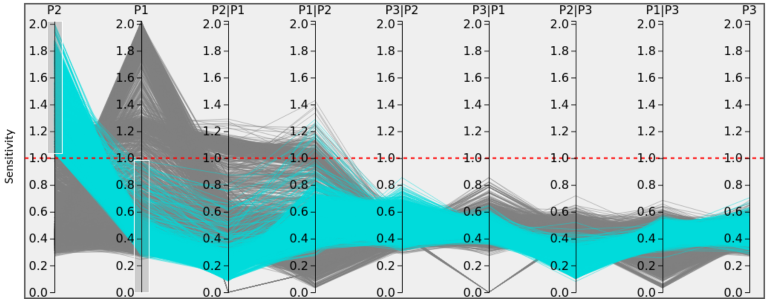


Figure 7.5: PCP allows for obtaining an overview of the sensitivity values. Each axis shows the sensitivity to one parameter setting, and each polyline represents one voxel. In this example, DGSA is used where values larger than 1 indicate a sensitivity to the corresponding parameter. This threshold is shown as a red dashed line.

Each sensitivity volume representing the sensitivity to a single parameter is shown as one axis. Thus, we obtain one axis per input parameter. Each spatial sample is shown in parallel coordinates as one polyline. To allow interactions with the PCP at interactive rates and data with higher spatial resolutions, we apply a Monte Carlo subsampling. Then, each polyline in the PCP represents one of the spatial Monte Carlo samples.

The horizontal screen space limits the scalability of PCP with the number of dimensions. Therefore, we support horizontal scrolling to allow for higher numbers of axes. However, many axes without a pre-defined order are still cumbersome to analyze. Therefore, we allow for reordering the axes based on the average sensitivity of the voxels. Thus, the axis showing the parameter with the highest sensitivity is shown first. Depending on the analysis task or for a more detailed analysis, some parameters might be irrelevant to the current analysis. To reduce the dimensionality, it is possible to filter out irrelevant parameters. We allow filtering based on a user-defined threshold for the average sensitivity value.

The quantitative sensitivities of the parameters can be directly compared by observing the PCP. To facilitate comparative analysis, we scale all axes equally. This also avoids misinterpretations, as all axes are comparable. For normalized sensitivity measures such as δ or Sobol indices, the axes can also be scaled to the range $[0, 1]$. The sensitivity computed by DGSA allows for differentiating whether a voxel is sensitive or not depending on the sensitivity value exceeding the threshold of 1. To facilitate the visual analysis in the PCP, this threshold of 1 can be visually conveyed by a dashed red line as shown in Figure 7.5.

Depending on the application domain, linking the sensitivity values to other data might be desirable. For example, in the medical context, one might want to include the ablation probability in the analysis directly. Another option is to include derived data like the mean field. For this purpose, we support additional domain-specific information in the parallel coordinates. For visually differentiating the additional axes from the sensitivity axes, the sensitivity axes are placed inside a gray box while the additional axes are rendered outside.

Besides getting an overview of the different sensitivity values, PCPs also allow for analyzing the correlations between the sensitivities to different parameters (see task T7.3). Horizontal patterns indicate positive correlations, while crossing patterns appear for negative correlations between neighboring axes. Interactively varying the order of the axis allows the investigation of the relations between different parameters. More subtle patterns can be observed by brushing on the axes.

However, PCPs do not contain the spatial information of the underlying volumes, but investigating the spatial positions of highly sensitive voxels is of high interest. We achieve this goal by linking the PCP to a surface rendering that shows the surface of the voxels in the intervals selected by brushing on the axes.

7.4.2 *Spatial Sensitivity Visualization*

PCPs allow for an exploration of the distributions of the sensitivity values. By brushing and linking to a 3D visualization, it is also possible to investigate spatial variations. However, a lot of interaction is necessary to fully investigate the distributions in multiple fields based on the spatial positions, which can become very time-consuming. Therefore, we provide a global spatial overview visualization (task T7.2). This allows for identifying regions where most parameters are sensitive and those with sensitivities to single or no parameters.

For this spatial overview, we use space-filling curves (SFC) to create a 1D linearization of the 3D volumes. We can then show the sensitivities directly over the SFC. This allows for reading off the different sensitivity values and spot regions of interest while providing spatial context by the SFC. We will start by explaining the computation of the adapted SFC before describing the visual design to visualize multiple sensitivity fields.

7.4.2.1 *Space-filling Curves for Multi-field Data*

The algorithm by Zhou et al. [399] is targeted at single field data. If we used one of the sensitivity volumes for the SFC computation, only the features of this specific volume would be preserved. However, we cannot assume that our sensitivity volumes are sufficiently similar; thus, features in other volumes would not be preserved well. Therefore, we generalize the value coherency term $N(C_i, C_j)$ (see Equation 7.2) to consider all fields while the locality preservation term can be maintained.

We need to choose a suitable distance measure between two grid points to adapt the value coherency. On each grid point x_i, the vector $\mathbf{s}_i$ contains all sensitivity values. Different options for computing the distance $d(\mathbf{s}_a, \mathbf{s}_b)$ between two vectors $\mathbf{s}_a$ and $\mathbf{s}_b$ exist. In this work, we investigate the following options, which we evaluate in Section 7.5.3:

1. *Sum of distances (*L_1*-norm)*: The first option is generalizing the originally proposed measure by summing up the distances for each field. This results in

   $$d_1(\mathbf{s}_a, \mathbf{s}_b) = \sum_i |s_{b,i} - s_{a,i}|,$$

   where $s_{a,i}$ and $s_{b,i}$ are the i-th component of $\mathbf{s}_a$ and $\mathbf{s}_b$, respectively. This also corresponds to the distance measure proposed by Dafner et al. [74] for RGB images.

2. *Euclidean distance (*L_2*-norm)*: The Euclidean distance is one of the most common measures between two vectors and is defined as

   $$d_2(\mathbf{s}_a, \mathbf{s}_b) = \sqrt{\sum_i (s_{b,i} - s_{a,i})^2}\,.$$

3. *Sum of squared distances*: As computing the square root for the Euclidean distance might be computationally expensive, one often considers the sum of squared distances, which are given as

   $$d_3(\mathbf{s}_a, \mathbf{s}_b) = \sum_i (s_{b,i} - s_{a,i})^2\,.$$

4. *Cosine distance*: The cosine similarity is a common similarity measure between vectors. When mapping the similarity to normalized distances, it can be written as

   $$d_4(\mathbf{s}_a, \mathbf{s}_b) = 1 - \frac{\sum_i s_{b,i} s_{a,i}}{\sqrt{\sum_i s_{b,i}^2}\sqrt{\sum_i s_{a,i}^2}}\,.$$

> Note that the cosine similarity in our case lies in the range $[0, 1]$ instead of $[-1, 1]$ because all sensitivity values are supposed to be positive.

The distance measures d_1 to d_3 are also normalized to $[0, 1]$. The distance measure for single fields, as used by Zhou et al., is included as a limit case in the definitions of d_1 and d_2.

7.4.2.2 *Combined Horizon Graph Visualization*

The sensitivities for the different fields can be shown over the SFC. The number of spatial samples usually exceeds the number of pixels available on standard screens. Weissenböck et al. [370] use a nonlinear scaling based on the variance of the data to reduce the space needed for visualization. However, in our visualization, it is unclear which regions are of higher interest than others and, thus, should be preserved by a nonlinear scaling. Instead, we use a spatial subsampling with the same sample points chosen for the PCP. While the SFC is computed on the entire sensitivity visualization, we only use the subsamples for rendering. If a specific region of interest is identified, the users can still zoom into the corresponding interval of the SFC and perform a more detailed analysis.

We considered several different design alternatives for the visualization of the sensitivity values. One option is table-based visualizations, where each volume is represented by one row, and columns represent the voxels in the SFC order. However, reading off exact values in approaches such as heatmaps is difficult. While a TableLense approach [280] would alleviate this problem, it does not scale to the high number of voxels in a 3D volume. Another option is the visualization of all fields together in a scatterplot or line plot, but the discrete scatterplot does not represent the spatial coherency that we assume along the SFC and quickly contains visual clutter when analyzing multiple fields.

The line plot can be well justified by the assumption of continuous variations. However, rendering multiple line plots together is challenging to interpret due to overlap. Also, our data contains many fluctuations, which worsens this issue. For better interpretability, we color the areas below the lines and create a drawing order based on the sum of the sensitivity values. Using this order, the lines of the sensitivity volume with highly sensitive voxels are drawn in the back leading to less overlap with those that contain less sensitive voxels. Additionally, this ordering is consistent with the default ordering of the PCP. A solid red line can be drawn at the threshold value for better interpretability of sensitivity values with a clear threshold, such as DGSA. Figure 7.6 shows one example for using multiple line lots to show the sensitivities.

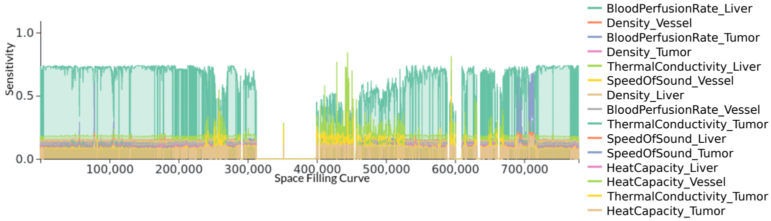


Figure 7.6: The sensitivity values can be shown over the SFC using a line plot where we draw the area below the curve and use a drawing order based on the sensitivities. However, this visual encoding quickly leads to overplotting.

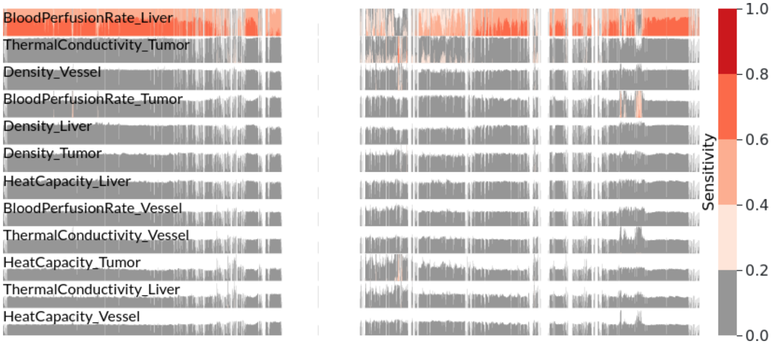


Figure 7.7: Showing multiple juxtaposed Horizon Graphs allows for a decluttered visualization but also requires a large amount of vertical space.

However, showing many sensitivity volumes together in a line plot still leads to overplotting. An alternative would be showing the different sensitivity volumes in several juxtaposed line plots that are stacked vertically. To reduce the required screen space in the vertical direction, we visualize the sensitivity values in Horizon Graphs [283], which were initially developed to visualize many time series. In the original approach, the number of bands is fixed while the range is adapted to the data. However, for our purpose, it is more meaningful to fix the height of the bands. The height depends on the choice of the sensitivity measure. For DGSA results, we fix the band height to 1 such that all insensitive values are shown in the same color. For the other sensitivity measures in the range $[0, 1]$, we choose a band height of 0.2, leading to 5 bands maximum.

The first band is colored gray. For DGSA, this band corresponds to the non-sensitive values, and for the two other sensitivity measures, it

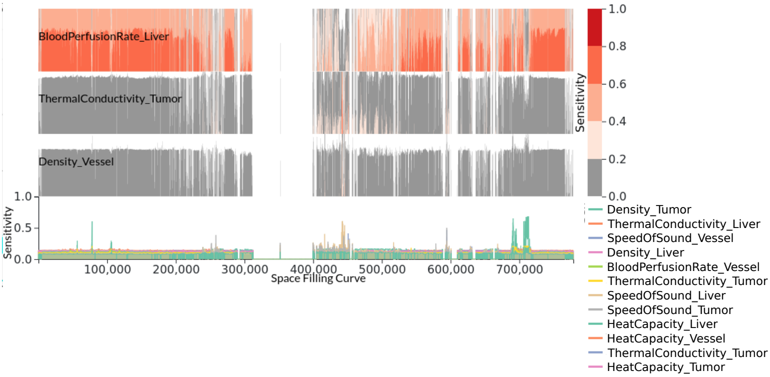


Figure 7.8: Combining Horizon Graphs with a line plot allows for a trade-off between the properties of the different visualizations.

corresponds to sensitivity values $\leqslant 0.2$, which can in general be considered less sensitive even though it is not a general threshold. We use discrete samples of a continuous white-to-red color map for the values higher than these thresholds. Thus, more reddish colors indicate higher sensitivity values. An example of multiple Horizon Graphs is shown in Figure 7.7. This visualization allows for reading off the values directly while also reducing clutter. However, a large amount of vertical space is required, which is not always available. The individual plots often become very small as we aim to include the visualization into a visual analysis tool with multiple coordinated views.

We combine the two visualization approaches to obtain a trade-off between their advantages and drawbacks, as shown in Figure 7.8. While Horizon Graphs require much vertical space, they do not suffer from overplotting. The line plots use the available space efficiently but overplotting hinders the interpretability. Therefore, the first q sensitivity volumes are shown as Horizon Graphs where the user can interactively define q. The remaining sensitivity values are shown in a line chart. As we maintain the default ordering based on sensitivity, the sensitivity volumes of the most sensitive parameters are shown as Horizon Graphs, while the less sensitive volumes are combined in the line chart. Reordering the PCP axes also influences the order in this visualization and, thus, provides coherency between the two visual encodings.

Similar to the PCP, we allow for filtering based on the sensitivity. Additionally, it is possible to brush in the spatial sensitivity visualization. The polylines in the PCP that correspond to the selected spatial positions are highlighted. The spatial positions are shown in the surface visualization. Even though the SFC provides spatial information, linking to the 3D domain is important to understand the spatial context intuitively.

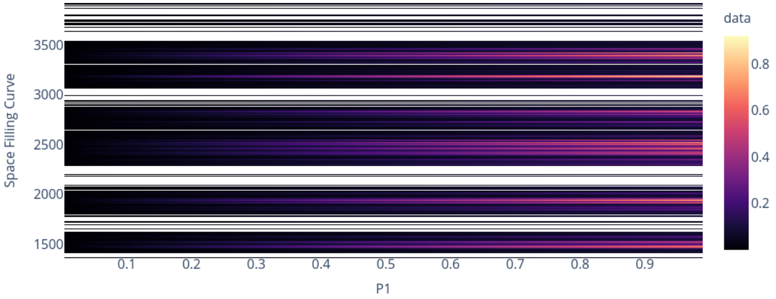
7.4.3 *Parameter Dependency Visualization*

The PCP and the spatial sensitivity visualization allow for identifying spatial regions of high sensitivities. The next goal is identifying the qualitative dependency of the simulation outcome on the input parameter (task T7.4). For this purpose, we encode the simulation output in a heatmap. Each row represents a single voxel where the rows are ordered according to the SFC. The columns represent the parameter values. While using the columns to encode the voxels would provide a direct relation to the spatial sensitivity visualization, we aim to show the change in the simulation output with the variation of the parameter values. For this purpose, showing the varying variable in the horizontal direction is more intuitive [377].

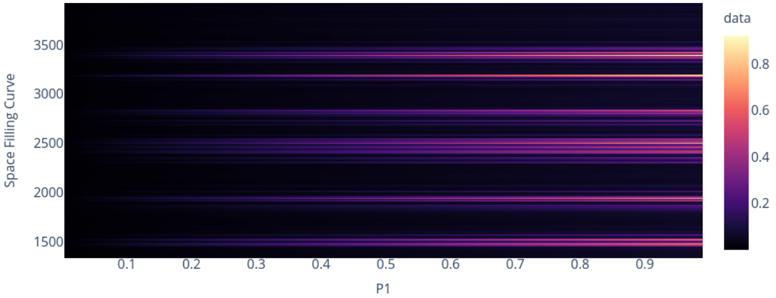
We use the same subsampling to reduce the amount of data and be consistent with the other visualizations. Further, we create a grid for aggregating the data. For each grid cell, we compute the mean and aggregate over the other parameter values not directly shown in the heatmap. This procedure tends to smooth the data and also removes outliers. However, if the grid resolution is high enough, the overall trends are captured sufficiently well. A grid resolution of 150×500 is used for all examples in this chapter. We choose a lower resolution in the x-direction because the number of simulation runs and, thus, the number of parameter-space samples is significantly smaller than the number of spatial samples.

For color-coding the cells in the heatmap, we use the perceptually uniform `magma` color map provided by matplotlib [350]. In the case of empty cells in the heatmap, which can occur either if the parameter space is not sampled densely or when selecting spatial regions in the linked visualizations, we visualize gaps in the heatmap as shown in Figure 7.9. This allows users to see and consider the gaps in interpreting the data. At the same time, the gaps in the visualization might complicate the analysis. We address this issue by allowing the user to switch to a dense visualization in which nearest neighbor interpolation is used to fill the gaps, see Figure 7.9a for an example. Interactively switching between visualizations facilitates understanding and interpretation while still keeping the awareness of missing data.

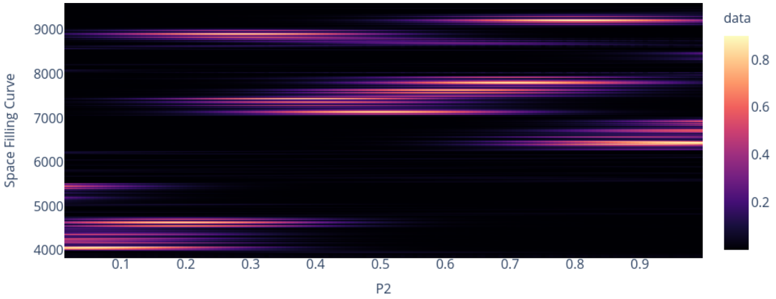
The proposed visualization for the parameter dependency allows for identifying different features. One of them is changes in the data as shown in Figure 7.9b. In this example, we see that the simulation output increases with an increase in the parameter values. Our visualization



(a) Missing data encoded by gaps.



(b) Nearest neighbor interpolation.



(c) Motion of spatial regions.

Figure 7.9: The parameter dependency visualization shows the simulation output as colors in a heatmap over the parameter values and the SFC. a) Gaps in the heatmap indicate missing values but complicate the interpretation. b) For an undisturbed analysis, the gaps can be filled by nearest neighbor interpolation. c) Besides changes in fixed positions, it is also possible to visualize spatial variations driven by parameter changes.

allows for estimating the spatial extent of this feature by considering the number of heatmap cells that belong to it, as shown in Figure 7.9a. Note that the size estimate is only meaningful when not using the interpolation. However, one drawback of this visualization is the difficulty of analyzing the nature of the increase in more detail which would be easier using a line chart. Another feature that could be identified in our visualization is the motion of spatial regions driven by parameter variation. The example shown in Figure 7.9c corresponds to a Gaussian that moves across the volume. This is indicated by the high values visible in different spatial positions. The feature is split into different regions of the SFC. The SFC was computed based on the sensitivity values, while the parameter dependency visualization shows the actual data values which were not considered when defining the SFC. Nevertheless, the feature remains visible in the visualization.

7.5 EVALUATION

In the following, we evaluate the choice of the distance measure for computing the space-filling curve, followed by a comparison of the sensitivity measures considered in this work. While we evaluate these two algorithmic components based on quantitative quality metrics, the utility of the visual design is shown in Section 7.6.

7.5.1 *Datasets*

We verify and evaluate our approach based on three datasets. At first, we use a synthetic dataset with known ground truth. Additionally, we use two simulation ensembles from the medical domain.

7.5.1.1 *Synthetic Data*

To verify the computation of the sensitivity volumes, we create a synthetic dataset. The ensemble consists of 4096 members with a spatial resolution of $32 \times 32 \times 32$. Each ensemble member depends on the three parameters a_1, a_2, and a_3 that range from 0 to 1. While parameter a_3 does not influence the result, the other two parameters influence three Gaussian kernels. The scalar field $g(x)$ for each ensemble member can be computed as

$$\begin{aligned} g(x) &= P_1 \cdot f(x;(7,7,7),3) + P_1 \cdot P_2 \cdot f(x;(10,25,15),3) \\ &\quad + f(x;(20,20,5+P_2 \cdot 20),3) + \zeta, \end{aligned}$$

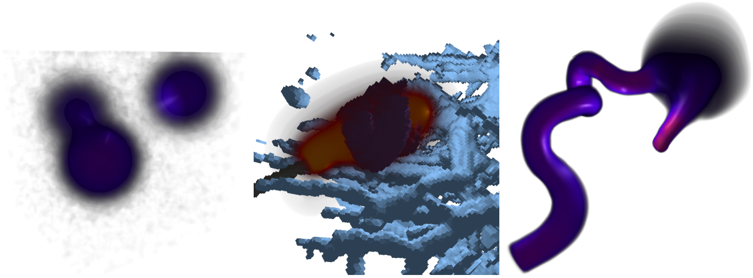

(a) Synthetic. (b) Radiofrequency ablation. (c) Blood flow.

Figure 7.10: Different simulation ensembles are used to evaluate our approach. a) Gaussian kernels define the synthetic dataset. b) Showing the magnitude of the flow created by a hemodynamics simulation reveals a complex set of patterns. c) The temperature field in radiofrequency ablation simulations (red/yellow volume rendering) together with the underlying tissue (blue: vessels, violett: tumor) is visualized in a volume rendering.

where $f(\mathbf{x}; (x_1, x_2, x_3), \sigma)$ is a 3-dimensional Gaussian kernel with standard deviation σ centered at (x_1, x_2, x_3) and ζ denotes uniform random noise in the range $[0, 0.01]$. A volume rendering for a simulation member is shown in Figure 7.10a.

7.5.1.2 *Radiofrequency Ablation*

For the first real-world dataset, we choose radiofrequency ablation simulations. Radiofrequency ablation is a minimally invasive treatment used to ablate tumors by heating the tissue. Simulations are used to predict the percentage of ablated healthy and malicious tissue. However, the tissue properties that are needed as input parameters are highly patient-dependent and hard to measure but also highly influence the amount of ablated volume [310, 3]. Therefore, simulation ensembles are created to cover the uncertainty introduced by the choice of input parameters. As the number of input parameters might be high, a parameter-space analysis can be used to find the most relevant parameters for the variation in the output data. Sandeep Gyawali and Tobias Preusser (Fraunhofer MEVIS, Jacobs University Bremen) provided the simulation tool for creating the dataset described in this section. An analysis of the parameter dependencies was performed by Heimes et al. [151]. When analyzing this dataset, we will compare our findings.

This chapter considers a radiofrequency ablation simulation ensemble consisting of 1024 simulation runs. Each ensemble run contains only a single time step representing the temperature field with a resolution of $92 \times 92 \times 92$. Figure 7.10b shows an example of an ensemble member. The simulation outcome depends on 5 different tissue properties, which are the tissue density, the heat capacity, the thermal conductivity, the blood perfusion rate, and the speed of sound, where each one varies for the three different tissue types liver, tumor, and vessel. Thus, we obtain a 15-dimensional parameter space.

7.5.1.3 *Blood flow simulations*

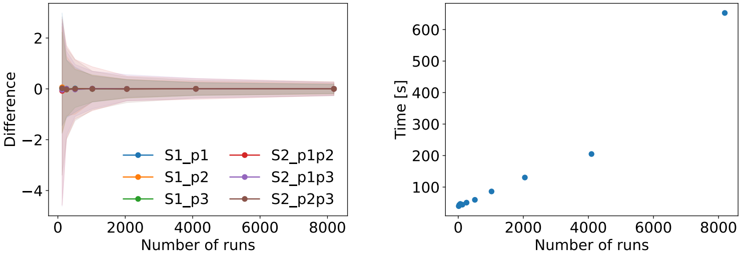
As an additional example, we consider blood flow simulations. While the goal is similar to the dataset used in Section 4.8.1, this dataset has a different geometry and input parameters. Here, we consider an aneurysm as shown in Figure 7.10c. The simulation model is driven by four different parameters. The viscosity and the density of the blood influence its flow properties. The third parameter is the maximal flow velocity of a parabolic flow profile which is sampled at the inlet of the simulation domain. The simulation is a so-called large eddy simulation that follows a turbulence model. This model is driven by the so-called Smagorinsky constant, which is a dimensionless model parameter.

The simulation ensemble consists of 320 simulation runs with a spatial resolution of $257 \times 119 \times 128$. In this chapter, we consider the flow magnitude as a scalar field and aim to investigate the input parameters' influence.

7.5.2 *Comparison of Sensitivity Computation Methods*

First, we want to compare different sensitivity computation methods. While many other methods exist, we focus on the ones presented in Section 7.1.1. We want to compare the sensitivity metrics concerning convergence depending on the number of runs and computation times. Finally, we will provide a visual comparison based on single slices and the dependency on single parameters. We use the synthetic dataset presented above for this evaluation but vary the number of samples. We sample the parameter space using Saltelli sampling [297], which leads to 16 to 8192 samples.

To compare the sensitivity measures, we compute the spatial sensitivities for all three measures. To investigate whether the sensitivity indices converge, we compute the difference of the sensitivity values for each voxel to those of the same voxel but with fewer runs. Thus, we obtain the differences when increasing the number of runs. These values are aggregated over the sensitivity volumes by plotting the mean of the vol-



(a) Convergence.

(b) Timings.

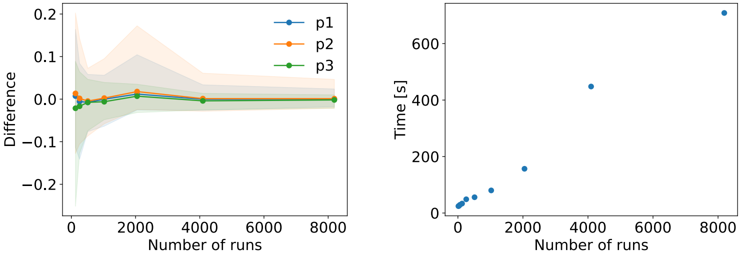
Figure 7.11: The variation of the Sobol sensitivities decreases with an increasing number of simulation runs (a), and the computational costs increase (b).

ume. To show the variation, we show the range from the minimum to the maximum as a band surrounding the mean.

Figure 7.11 shows the results for Sobol sensitivity indices. We observe a significant variation for less than 1000 values that significantly decreases for larger numbers of samples. However, the computational costs also increase significantly, where we observe an approximately linear trend.

When observing the results for the δ sensitivity shown in Figure 7.12, a decreasing trend can also be observed but is weaker than for the Sobol indices. Additionally, there is an increase in the variation for 2048 runs. However, the variation is significantly smaller compared to the Sobol indices. Even though the values for both sensitivities cannot be directly compared, both measures span the range $[0, 1]$ and, thus, cover a comparable range. This indicates that the δ sensitivity measure is less sensitive to the number of runs. Even though the computation times are generally higher, they are in the same order of magnitude and increase approximately linearly with the number of simulation runs.

For the DGSA sensitivity measure, the results are shown in Figure 7.13. We cannot observe any decrease with an increase in the number of simulation runs, and the variation is very large. However, it needs to be taken into account that this sensitivity measure is not normalized between 0 and 1, but instead, sensitive voxels are larger than 1, where the exact values of the sensitivity measure are less intuitive to interpret. The computation times for DGSA are significantly larger and scale worse with increased numbers of ensemble members. However, DGSA also depends on additional parameters. For a detailed analysis, see Appendix A.3.



(a) Convergence.

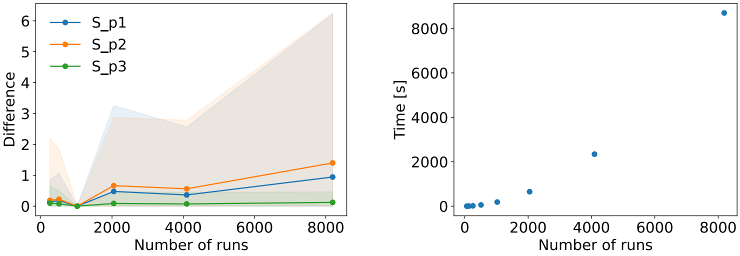
(b) Timings.

Figure 7.12: The variation of the δ sensitivities decreases with an increasing number of simulation runs, but for 2048 runs, very high values can be observed (a). For the computational cost, we also see an approximately linear increase (b).

They are also compared visually to obtain a better understanding of the different sensitivity measures. Figure 7.14 shows a single slice of one sensitivity volume for the different datasets. We do not consider the DGSA sensitivity measure for the ablation and aneurysm datasets, as this method requires high computation times. For the Sobol sensitivity measure, we only consider the first-order Sobol indices.

For the synthetic dataset, we observe that similar features are detected by all sensitivity measures. Most prominently, the peak in the upper left, which is strongly influenced by the parameter P_1, whose sensitivity is shown in the images, is detected by all methods. While Sobol sensitivity and DGSA correctly identify the horizontal region as insensitive to P_1, the values for δ are slightly increased. For Sobol indices, the noise in the background, where no features are present in the data, also appears in the sensitivity volumes. This can be explained by the definition of the Sobol indices, which are based on distributing the variance of the output to the different input parameters. However, as the random noise is independent of any input parameter, it cannot be attributed correctly.

In the case of the ablation dataset, we observe a central slice for the sensitivity to the blood perfusion rate of the liver. The Sobol indices for this volume are in the range $[-0.43, 1.45]$, which is significantly outside the range $[0, 1]$ in which the indices are supposed to be located. This is a very strong indicator that we did not use enough parameter-space samples to estimate the Sobol sensitivities correctly and, therefore, observe a numerical problem. However, considering the 15-dimensional parameter space, choosing enough samples is computationally costly. The high number of simulation runs also agrees with common criticism of the Sobol sensitiv-



(a) Convergence (number of runs).

(b) Timings (number of runs).

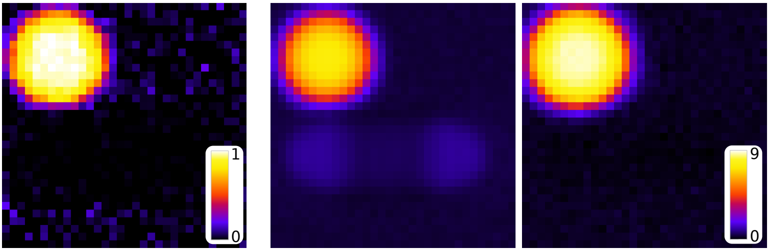
Figure 7.13: For the DGSA sensitivity measure, no convergence with a larger number of samples can be observed (a). The computational costs of the method scale worse than those for the other techniques (b).

ity analysis, for example, Saltelli et al. [298] propose $17,000$ simulation runs for 15 input parameters. The result for the δ-sensitivity measure seems significantly more plausible. The general structure of the spatial distribution shows smaller values on the boundaries of the domain, as we can also observe for the Sobol index.

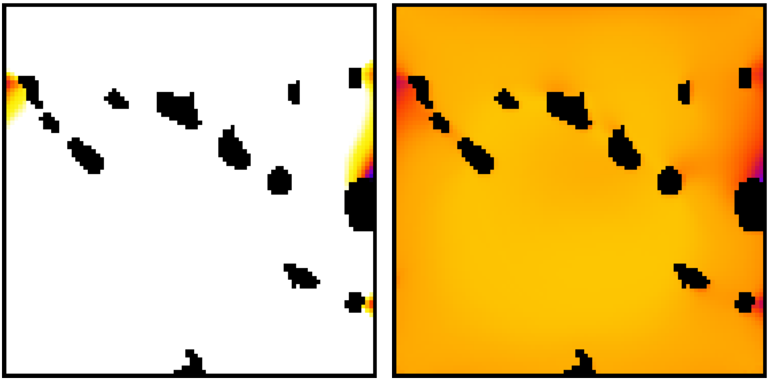
For the aneurysm, the Sobol sensitivity indices are often outside the range $[0, 1]$ indicating too few samples. We generally observe a similar spatial structure for the δ sensitivity measure, even though the shapes between the less sensitive regions vary slightly. The region highlighted by the green box in Figure 7.14g shows no variation for Sobol indices.

The spatial structure is generally similar between the different datasets. Every sensitivity method has their advantages. While Sobol indices are widely spread, and the analysis of variances is intuitive to interpret, they require a special sampling scheme. Even though we followed the efficient Saltelli sampling [297], our number of simulation runs is significantly too small. Additionally, the results for Sobol indices are not meaningful if the simulation parameters are not the only factor influencing the simulation output. Besides random noise, as in our synthetic dataset, this also applies to initial conditions, which are often varied for simulation ensembles.

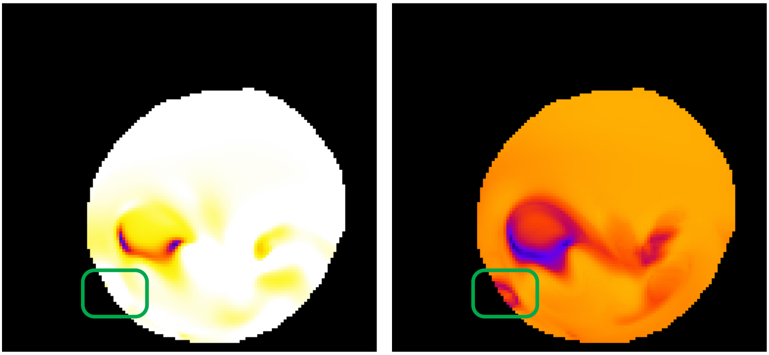
The δ sensitivity measure generally provides smooth and plausible results even though the horizontal structure visible in Figure 7.14b does not reflect a region sensitive to the analyzed parameter. However, the variations are minor. Like the Sobol indices, this sensitivity measure can be computed with reasonable computational costs. Using 32 cores on the PALMA high-performance computer, the computations for the δ sensitivity took around 45 min.



(a) Synthetic, Sobol. (b) Synthetic, δ. (c) Synthetic, DGSA.

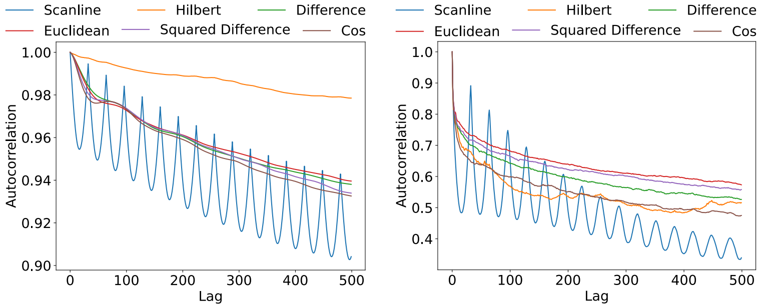


(d) Ablation, Sobol. (e) Ablation, δ.



(f) Aneurysm, Sobol. (g) Aneurysm, δ.

Figure 7.14: We show the same slice of a sensitivity volume for different sensitivity measures. For the synthetic dataset, we show the sensitivity to P_1 (a-c), for the ablation dataset, we show the sensitivity for the blood perfusion rate of the liver (d-e) and for the aneurysm, we show the sensitivity to the inlet velocity (f-g). The green box points out a region where the structure varies between both sensitivity methods.



(a) Positional coherency.

(b) Value coherency.

Figure 7.15: The autocorrelations for the space-filling curves allow comparing the different methods. For all results, all three datasets and the Sobol and δ sensitivity indices are considered.

The computation times for DGSA are significantly higher, and the numerical analysis showed that it does not converge with an increasing number of samples. Therefore, we omitted DGSA in the comparison for real-world data, even though results for the synthetic dataset do not show any undesired artifacts. An advantage of DGSA is that it only depends on the distances between the simulation results on the respective spatial positions. The generalization to temporal data is straightforward by choosing a suitable distance measure, which is not the case for the other sensitivity measures. However, due to the disadvantages, we only recommend applying DGSA if the other methods are unsuitable.

For the usage scenarios in this chapter, we choose the δ sensitivity measure because it can be computed in reasonable times and requires fewer simulation runs than the Sobol indices. They are not applicable to the available data without running significantly more simulations or defining a surrogate model. Additionally, the computation of δ does not require a specific sampling scheme for efficient computation, such as Sobol indices. Therefore, we consider this sensitivity measure as more generally applicable in the field of spatio-temporal simulation ensembles.

7.5.3 *Space-filling Curve*

Similar to previous works [399, 74], we evaluate our space-filling curves using autocorrelation. For the measures of which the autocorrelations are computed, we follow the definitions by Zhou et al. [399]. As a first measure, we use the autocorrelation of the linearized data values. However, considering multi-field data, we compute the autocorrelations for each field individually and average over all fields. As a second measure, we use the autocorrelation of radial Euclidean distances where we can directly apply the proposed definition given as $t(i) = \left\|\mathbf{p}_i - (0,0,0)^T\right\|$, where $\mathbf{p}_i$ denotes the spatial position of the i-th point on the curve.

To obtain a guideline on which algorithm to use, we compare the data-driven SFCs with the four different distance measures but also include the scanline algorithm providing a trivial linearization and the Hilbert curve [157]. Those two SFCs do not consider the underlying data. For a more efficient and comparable evaluation, we resized all datasets to $32 \times 32 \times 32$. For each SFC, we considered the three datasets and their Sobol sensitivity volumes as well as the δ sensitivity volumes. The autocorrelations are computed separately and then averaged for each SFC computation method.

The results are shown in Figure 7.15 (see Appendix A.4 for results for the individual datasets). The scanline approach provides the worst coherency for both criteria. The Hilbert curve performs best with regard to positional coherency (see Figure 7.15a), which is to be expected due to its definition. The data values are not considered when creating the curve. The positional coherency is comparable between the four different data-driven SFC approaches. However, the differences in the value coherency are larger. There are differences in the best distance measure between the different datasets, as seen in the separated results presented in Appendix A.4, but they are relatively small. On average, the Euclidean distance provides the best value coherency while preserving the locality best among the data-driven methods. Therefore, it provides a good compromise between those two optimization goals and will be used for the remainder of the work.

7.6 USAGE SCENARIOS

In the following, we present the analysis of the synthetic dataset (described in Section 7.5.1.1) to verify our visual encodings. Then, we analyze two real-world datasets to show the general applicability. We use the δ sensitivity measure and data-driven SFC with Euclidean distances for all datasets.

7.6.1 *Synthetic data*

First, we use the PCP to obtain an overview about the general sensitivity values. As shown in Figure 7.16, we observe that the parameter P_3 does not influence the result as the sensitivity values are very small.

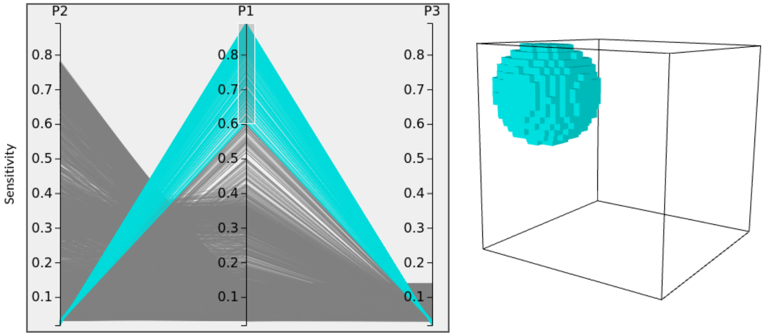


Figure 7.16: The PCP show that the simulation output is not sensitive to P_3. When selecting high values of the sensitivity to P_1, the surface visualization shows that the corresponding voxels belong to one of the Gaussians.

This meets our expectations based on the definition in Equation 7.3. The other two parameters influence the result. For a better understanding of the influence, we brush in the PCP to select a range of high sensitivites in parameter P_1. The kind of dependency can be investigated in the parameter dependency visualization shown in Figure 7.9. Here we see the linear increase in the simulation output which confirms our expectations.

To investigate the spatial variation in more detail, we use the spatial sensitivity visualization shown in Figure 7.17. Here, we observe regions sensitive to P_1, P_2, and both. We start by selecting the spatial region which is sensitive to both parameters. As expected, the corresponding voxels are located around the Gaussian, whose height is driven by the product of the parameters. The parameter dependency visualization reveals the expected linear increase in both parameters. When selecting the range that is only sensitive to P_2, the parameter dependency visualization shown in Figure 7.9c is obtained. As explained in Section 7.4.3, it shows the expected spatial variation of the output data. To summarize our results, all findings for the synthetic dataset agree well with its definition.

7.6.2 *Radiofrequency ablation data*

As the ablation dataset depends on 15 input parameters, it is important to identify the most influential ones. Because the ablation treatment aims to destroy only the tumor, it is crucial that we identify the simulation

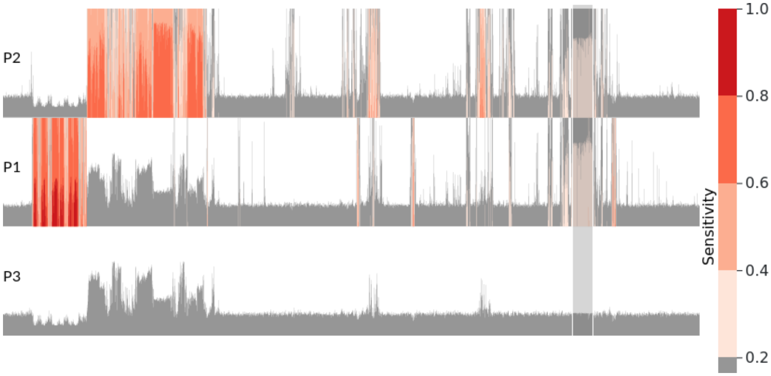


Figure 7.17: The PCP shows that the simulation output is not sensitive to P3. When selecting high sensitivity values to P1, the surface visualization shows that the corresponding voxels belong to one of the Gaussians.

parameters contributing most to the temperature uncertainty, especially in the region close to the tumor. While the tumor should be fully ablated, as little healthy tissue as possible should be harmed. To obtain an overview, we start with the PCP visualization. We can identify that for several parameters, no significant sensitivity is detected, so that they can be excluded from future analysis.

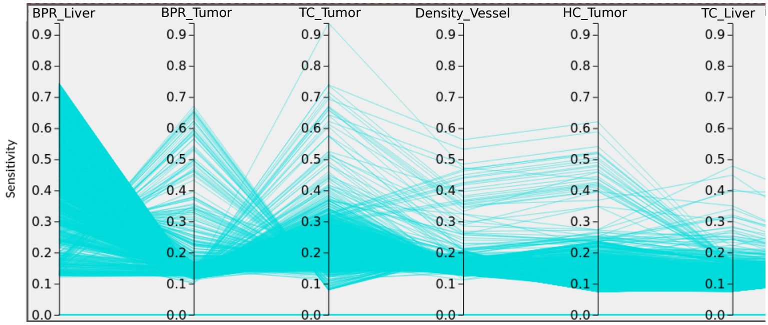


Figure 7.18: The PCP for the ablation dataset shows sensitivities to the blood perfusion rate (BPR) and thermal conductivity (TC) of the liver and tumor and the density of the vessel. Further parameter axes can be accessed by scrolling in the PCP. A correlation between the vessel's density and the tumor's heat capacity can be observed.

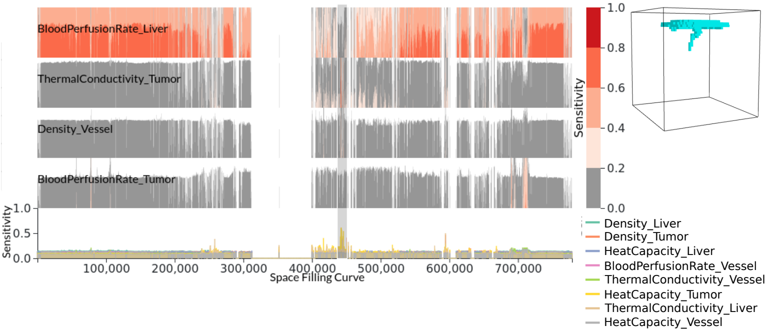
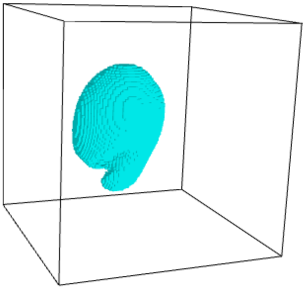


Figure 7.19: A part of the SFC which shows comparably small sensitivity values of the blood perfusion rate of the liver is selected. Even though the tumor's thermal conductivity and heat capacity are increased in this region, the 3D visualization reveals that this selected region belongs to the boundary of the liver.

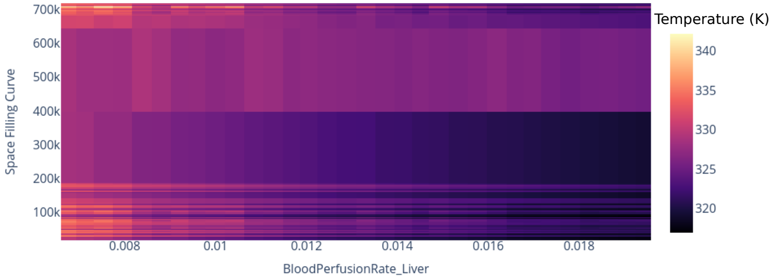
PCP axes for the most influential parameters are shown in Figure 7.18. We see that, by far, the liver's blood perfusion rate (BPR) is the most significant parameter, but also the BPR of the tumor is significant in various regions. By following the polylines between the axes, we can see that those voxels that are sensitive to the tumor's BPR are not the same as those sensitive to the BPR of the liver and the thermal conductivity (TC) of the tumor. Additionally, we observe that voxels with a high sensitivity to the density of the vessel are also highly sensitive to the heat capacity (HC) of the tumor. These two observations indicate that the tissue parameters do not only influence the simulation output in the corresponding tissue regions.

Investigating the spatial variation of the sensitivity reveals that the blood perfusion rate of the liver influences most regions. However, we also spot a region with relatively low sensitivity to the blood perfusion rate and increased values in the sensitivity to the thermal conductivity and heat capacity of the tumor. When selecting this region in the spatial sensitivity visualization as shown in Figure 7.19, the 3D surface rendering reveals that it is located at the boundary of the liver region and, thus, in a region of less interest for the analysis of this ensemble.

As this simulation ensemble mainly aims at investigating the ablation area, we include the probability of ablation in our analysis. Tissue can be considered ablated if the temperature is at least 327.15 K (54 °C). For each voxel, we compute the percentage of simulation members that exceed this threshold and, in the following, refer to it as the ablation probability. As the spatial regions not ablated in any of the simulation



(a) Surface visualization.



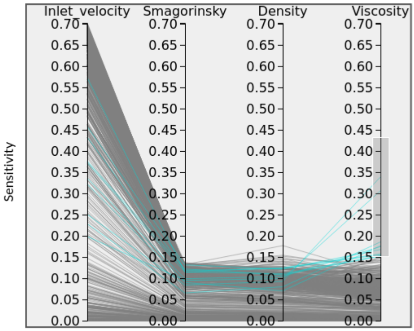
(b) Parameter dependency.

Figure 7.20: For a more targeted analysis, we only consider the voxels which are ablated in some ensemble members but not in all of them. The parameter dependency visualization reveals a temperature decrease with an increase in the liver's blood perfusion rate.

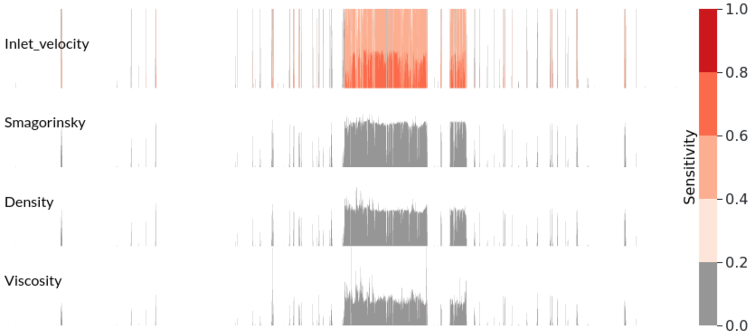
runs are of little interest, we select an ablation probability > 0 and < 1 in the PCP. The selected voxels are shown in the surface visualization in Figure 7.20a and belong to the boundary of the ablated area. The PCP reveals that the liver's BPR is the most influential parameter in this region. When investigating the dependency on this parameter, we obtain the result shown in Figure 7.20b. The visualization shows a clear temperature decrease with an increase in the liver's BPR. This can be interpreted as a higher blood perfusion rate causing a cooling effect in the tissue.

In general, the findings for this dataset confirm the findings of Heimes et al. [151], especially regarding the importance of the parameters. While their analysis mainly focused on the variation of the ablation area over a selected set of parameters, our approach allows a more general analysis that can also be applied beyond the ablation area.

7.6.3 *Aneurysm data*



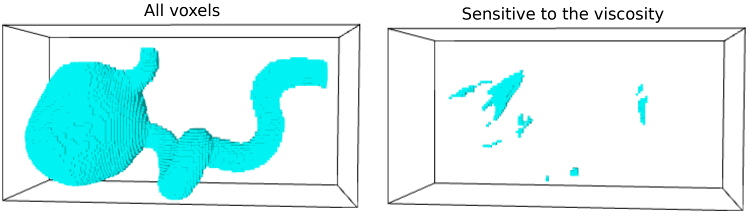
(a) PCP.



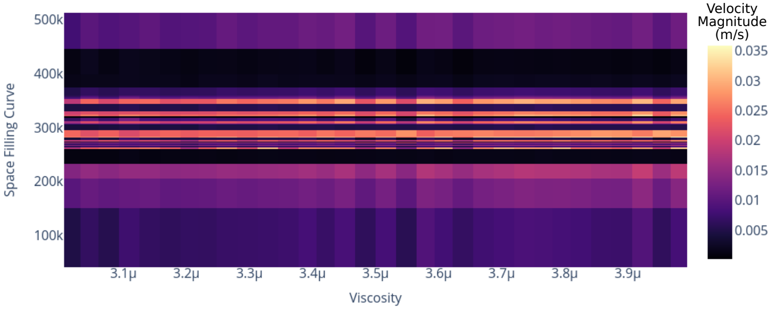
(b) Spatial sensitivity visualization.

Figure 7.21: The outcome of the aneurysm dataset is mainly sensitive to the inlet velocity, which can be seen in the PCP (a) and the spatial sensitivity visualization (b). The spatial sensitivity visualization also includes regions outside of the vessel which explains the large regions with a sensitivity of 0.

As a second example, we analyze the aneurysm dataset. To obtain an even number of samples in all dimensions, which is necessary for computing the SFC, we resample the dataset to a resolution of $128 \times 64 \times 64$. The PCP shown in Figure 7.21a reveals that the most influential parameter is the inlet velocity. Only the viscosity is influential in some regions, while the influence of the other parameters is negligible. The spatial sensitivity visualization shown in Figure 7.21b reveals that almost



(a) Surface rendering.



(b) Parameter dependency visualization.

Figure 7.22: The largest region of the voxels sensitive to the density is located in the aneurysm (a). The parameter dependency visualization (b) reveals only a minor increase but overall small velocity magnitudes.

all voxels that belong to the vessel or aneurysm are also sensitive to the inlet velocity.

To investigate the regions sensitive to density, we select the corresponding voxels in the PCP, see Figure 7.21a. The spatial positions are also sensitive to the inlet velocity. The 3D visualization in Figure 7.22a reveals that the spatial locations sensitive to the density are scattered across the domain. However, the largest connected spatial region is located in the aneurysm. This region can be related to the inflow region which is known to show a circulating structure of the flow and marks the transition from the laminar vessel flow to the more turbulent flow in the aneurysm [46]. When observing the parameter dependency visualization as shown in Figure 7.22b, we see only a minor increase depending on the viscosity. However, we can observe that the velocity magnitude is very small in these regions, reaching a maximum of $0.035\,\mathrm{m/s}$ where the maximum over the whole volume is $0.1\,\mathrm{m/s}$. Therefore, the observation of the increase should not be considered significant. In general, for this dataset, we can conclude that computations of additional simulations should focus on the inlet velocity, where the viscosity should also be considered as it influences on of the phenomena of interest. The influence of the other parameters is very small and they can be can be considered as less important for future analysis tasks assuming that the parameter range in this simulation dataset was chosen large enough.

7.7 DISCUSSION

This chapter presents a method for interactively analyzing spatial variations in parameter sensitivities. We include three different sensitivity measures, evaluate them and discuss their advantages and drawbacks. To visualize multiple sensitivity volumes while reducing occlusion, we propose to use an SFC to linearize the spatial domain. For this purpose, we include different options to generalize data-driven SFCs to multifield data and identify the Euclidean distance as the most promising distance measure. Based on these considerations, we propose an interactive visual analysis tool to investigate different aspects of parameter sensitivity.

Our approach scales well with the number of parameters, as we showed when analyzing the 15-dimensional parameter space. As discussed in Chapter 4, dense samplings of higher-dimensional parameter spaces for spatial simulation data are rare due to the high computational cost. To analyze the sensitivity to the input parameters, more samples in the parameter space are required than for partitioning it. In the case that the spatio-temporal simulation has more input parameters, it is possible to run a pre-screening and only include those parameters in the further analysis, to which the simulation output is most sensitive. Therefore, we do not consider the scalability with the number of parameters a problem. However, the scalability with the spatial resolution yields more challenges. Even though we partially alleviate this problem by applying a subsampling, the number of samples required to avoid a significant loss of information also grows the number of voxels in the original data.

We provided an algorithmic evaluation of the sensitivity computation and the SFC, and the applicability of our visual analysis tool is shown in three usage scenarios. However, a domain expert evaluation would be beneficial for more insights into the approach's utility. To better understand the spatial sensitivity visualization, performing a user study that compares the different designs might be beneficial. Also, alternative encodings of multi-field data exist, like multi-field volume renderings that could be included in a more in-depth investigation. However, they do not scale well with the number of volumes to visualize and quickly suffer from occlusion.

This chapter only considered a single scalar field for the simulation output. However, for a broader applicability of the analysis approach, an generalization to time-varying multi-field data would be desirable. The sensitivity computation using DGSA could be directly generalized by choosing a suitable distance measure for the clustering. For temporal data it was already discussed by Fenwick et al. [103]. Also, generalizations to multi-field data could be easily included. However, applying DGSA would also require a more efficient implementation to be applicable to real-world data without subsampling the spatial resolution. After sensitivity volumes are computed, the only visualization that needs to be adapted would be the parameter dependency visualization to work with the different simulation outputs.

In the future, sensitivity indices for the interactions between parameters could be included in the analysis. The correlations between the different sensitivities provide a first hint towards relationships, but including the statistical measures allows for a more in-depth analysis. While this is straightforward for lower-order interactions by including the additional sensitivity volumes, analyzing the qualitative parameter dependency remains an open challenge. Additionally, the number of sensitivity volumes grows significantly if higher-order interactions should also be analyzed. In this case, further adaptation to improve the scalability is necessary.

Overall, our interactive visual approach supports a comprehensive analysis of spatial variations in sensitivities to the input parameters. We derived guidelines for choosing a suitable sensitivity computation algorithm among three different options and generalized data-driven space-filling curves to multi-field data.

Part III

UNCERTAINTY-AWARE ANALYSIS OF CORRELATIONS

8

SIMILARITY IMAGES

Climate scientists are interested in studying time-varying phenomena on different spatial scales. By investigating relationships between different spatial regions, scientists obtain a deeper understanding of these phenomena. Computing correlations is a common approach to study these relations between different regions. In particular, homogeneous regions, where time series behave similarly, exhibit high positive correlations. Additionally, relations between more distant regions can be investigated by using correlation analysis. In this context, also negative correlations are of interest. For example, the pressure over Iceland and the Azores are anticorrelated, and this coupling, also called the North Atlantic Oscillation (NAO), heavily influences the weather over Europe.

Computing pairwise correlations between spatial regions leads to a high number of correlation values as the size of the correlation matrix scales quadratically with the number of samples. In analyzing simulation ensembles, the spatial resolution is normally high and adds the additional dimension of ensemble members that needs to be considered. Developing a visualization of this matrix that preserves the global structure but also takes the spatial locations of the samples into account is a challenging task. Therefore, a common approach is choosing a reference point from which the correlation to all other spatial locations is computed. While significantly reducing the number of correlation values, this also results in a significant loss of information.

In this chapter, we present *similarity images*, where we use distances between colors to encode correlations among the different spatial regions. We embed the spatial samples in a 3D space by mapping correlations to a distance measure. The resulting samples can then be mapped to colors. By applying the colors to the spatial positions, we obtain a color-coded spatial visualization that we can use as input to a segmentation algorithm. This segmentation leads to a significant reduction of the data complexity by creating homogeneous regions of similar behavior. The following analysis steps can be based on these segmentations, see Chapters 9 and 10.

In the following, we will provide an overview of related work in correlation analysis in Section 8.1 followed by a problem specification in Section 8.2. Our approach to computing similarity images is presented in Section 8.3 while Section 8.4 describes how hierarchical segmentation can be used to obtain homogeneous segments. After that, we validate our approach using synthetic data followed by an application to real-world data in Section 8.5.

The methods of this chapter are published in

> **M. Evers***, K. Huesmann* and L. Linsen, Uncertainty-aware Visualization of Regional Time Series Correlation in Spatio-temporal Ensembles, *Computer Graphics Forum* 40, No. 3: 519-530 (2021) (* The authors contributed equally.)

All authors contributed to discussing ideas, writing, and editing the manuscript. The results and implementation of this chapter based on this publication were created in equal parts by Karim Huesmann and me.

8.1 RELATED WORK

Correlations are the most commonly used statistical measure for analyzing time series and provide a similarity measure between the different time series [270, 114]. Approaches [301, 171, 60] for correlation analysis on spatial data compute the correlations among different fields in multi-field data. However, these approaches do not consider correlations among different spatial regions. Pfaffelmoser and Westermann [261] show that correlations represent local phenomena in uncertain fields. Correlations between different spatial regions are used in the context of climate data where so-called climate networks [2, 39] are created by computing the correlations among different spatial regions. Then, a network is formed where an edge corresponds to a correlation value that exceeds a certain threshold. Nocke et al. [242] present a survey about the state of the art technique in visually analyzing climate networks. However, they identify occlusion and visual clutter as common problems and mainly focus on improvements of node-link diagrams that also provide geographical context. Nocke et al. state that matrix-based visualizations do not convey geographic information even though enhanced node-link diagrams provide too much clutter for global correlation analysis. Climate networks have also been used to study correlations among spatial regions between two fields [75, 94], but they do neither investigate more than two fields nor consider ensembles as input data.

In machine learning, correlation clustering is a technique using pairwise correlations to find clusters and maximize the correlations inside of the cluster while minimizing the correlations to elements of other clusters [26]. Zhang et al. [395] propose to use a distance metric that combines correlations with distances and builds on a k-means clustering. Like our approach, Sukharev et al. [326] use a dimensionality reduction technique before clustering. However, they use PCA directly on the data points and, thus, do not consider correlations among the different samples. Liebmann et al. [207] used hierarchical correlation clustering, but their approach does not lead to spatially connected components.

Spatial clustering has been investigated in other contexts, for example, using probability density functions [42], self-organizing maps [13], and wavelets [380]. Nocke et al. [244] used clustering for data reduction in climate data before visualizing the clusters. Correlation cliques [197] describe regions of highly correlated points but are based on a seed point and, thus, do not form a global approach. Antonov et al. [17] study long-range interactions in climate data, but their approach also strongly depends on the choice of the seed point. Jänicke et al. [166] also use clustering to apply wavelet analysis to larger datasets. However, they neither work on correlations nor consider ensemble data.

Pfaffelmoser and Westermann [260] analyze global correlations similar to our work, but they do not consider temporal data. Additionally, they only work on two levels of detail which does not suffice for an in-detail analysis. However, none of the discussed approaches analyzes global spatial correlations in spatio-temporal simulation ensembles.

8.1.1 *Watershed Segmentation*

Many different segmentation algorithms exist. In the following, we provide a brief background on a watershed segmentation algorithm that can be used for obtaining a hierarchical segmentation. These class of algorithms is based on the concept of filling basins in a heightfield formed by the data, where the flow of water is simulated to separate the different segments. In this chapter, we use an efficient implementation [257] of a graph-based watershed algorithm that operates on edge-weighted graphs [238, 69].

Let $G = (V, E)$ be a graph with vertices V and edges E where each edge $e_{i,j} = \{v_i, v_j\}$ between vertices v_i and v_j has a weight $\omega_{i,j}$. Based on the edge-weighted graph, a binary partition tree can be computed. A binary partition tree is a data structure that provides a multi-scale representation of an image [296]. More concretely, in the algorithm used here, a binary partition tree by altitude ordering [70] is used where each node corresponds either to an edge in the minimum spanning tree (MST) of G or to a vertex $v \in V$. The root of the tree corresponds to the edge of the MST that was added latest. The tree itself provides a strict ordering of the edges of the MST, where the altitude of the edges in the tree defines the ordering. If the edges of G are either already sorted or can be sorted in linear time with respect to their weights, the computation complexity for the binary partition tree is $\mathcal{O}(|E| \times \alpha(|V|))$ where $\alpha(x)$ is the inverse of the single-values Ackermann function and grows extremely slowly with the argument x. To obtain a watershed hierarchy, the relevant watershed edges are identified. An edge of the binary tree is marked as a watershed edge if it merges two segments that belong to different minima. Thus, we can mark all edges as being watershed edges or not, which can be achieved in linear run time in the number of edges. The watershed hierarchy is obtained from the labeled graph by only keeping the watershed edges and removing consecutive tree nodes with equal altitudes. The nodes of the watershed hierarchy form the watershed basins and provide the lowest level of detail.

8.2 PROBLEM SPECIFICATION

In the following, we consider an ensemble consisting of m simulation runs with n spatial samples. The ensemble can contain M different fields. In contrast to the previous chapters, we do not view the individual ensemble runs as spatial fields over time but instead look at the individual spatial samples where each spatial sample contains a set of time series, one for each ensemble member. Thus, each spatial sample x_i, $i = 1, ..., n$ contains a time series as a function $s_{ikv}(t)$ over time t for each simulation run r_k, $k = 1, ..., m$ and each field $v = 1, ..., M$. Given a simulation ensemble, we want to visualize the global correlation structure and find homogeneous spatial regions. As the spatial complexity might vary significantly among the different fields, we treat each field individually. However, we want to capture the full variability of the simulation ensemble. Therefore, we aim at a visual representation for the correlations of each field of the spatio-temporal simulation ensemble.

8.3 SIMILARITY IMAGE

The computation of similarity images contains several steps shown schematically in Figure 8.1. First, we need to define how to compute a correlation among two ensembles of time series. After computing a correlation

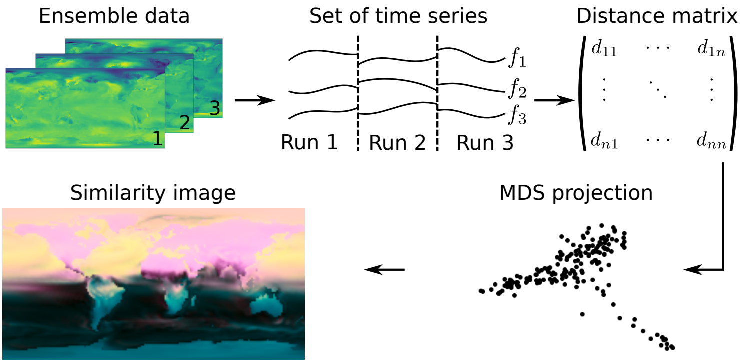


Figure 8.1: Overview of steps to compute similarity images from spatio-temporal ensemble data. The distance matrix based on correlations is used for an embedding, allowing for mapping positions to colors.

matrix between all spatial samples, we can use it to create a 3D embedding of the different spatial samples. Then, we map the embedding to colors such that the correlations are preserved as distances between the colors.

8.3.1 *Time Series Ensemble Correlation*

For computing the correlation among two sets of functions that belong to different spatial regions of the same ensemble, we only want to consider correlations among time series that belong to the same simulation run. One option would be to compute all correlations individually and then compute the mean among the correlation values. However, this leads to a significant loss of information. Instead, we propose to concatenate the individual time series and compute the correlation between the concatenated time series that represent the whole ensemble. The concatenated time series $f_{i\nu}(t)$ for the spatial sample $\mathbf{x}_i$ of field ν can be computed as

$$f_{i\nu}(t) = s_{i\nu k}\left(t - \sum_{l=1}^{k-1} T_l\right) \quad \text{for} \quad \sum_{l=1}^{k-1} T_l < t \leqslant \sum_{l=1}^{k} T_l\,,$$

where T_k is the number of time steps of the simulation run r_k.

These concatenated time series can be used to compute the correlation between the time series $f_{i\nu}(t)$ and $f_{jw}(t)$ between two spatial samples $\mathbf{x}_i$ and $\mathbf{x}_j$ of fields ν and w, respectively. We choose the Pearson correlation coefficient [32], which describes the linear dependency and is the most common correlation measure. We can compute the Pearson correlation $C_{iv,jw}$ between time series $f_{iv}(t)$ and $f_{jw}(t)$ as

$$C_{iv,jw} = \frac{\sum_p (f_{iv}(t_p) - \mu_{iv})(f_{jw}(t_p) - \mu_{jw})}{\sqrt{\sum_p (f_{iv}(t_p) - \mu_{iv})^2}\sqrt{\sum_p (f_{jw}(t_p) - \mu_{jw})^2}} . \tag{8.1}$$

Here, μ_{iv} and μ_{jw} are the mean of the concatenated time series $f_{iv}(t)$ and $f_{jw}(t)$. When computing the correlation matrix among all spatial samples of all fields, one would obtain a correlation matrix of size $(nM)^2$. An alternative is the computation of the correlation matrix $C_{iv,jv}$ for each field v individually, leading to M individual correlation matrices. While the first approach leads to consistent color coding among the different fields, the latter approach is computationally significantly cheaper.

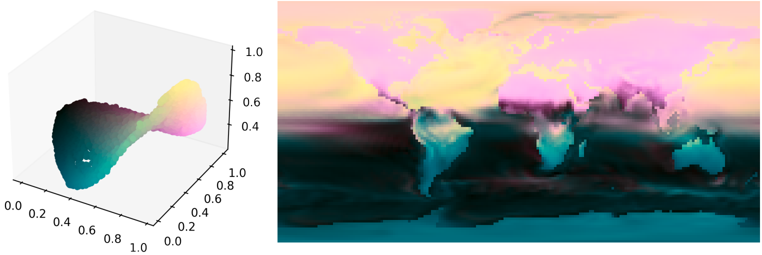
8.3.2 *3D Embedding*

To apply a color coding to the spatial samples, where distances between colors represent correlations, we need to define a color mapping. We, therefore, map the spatial samples to a 3D space based on their correlation, as color spaces are also three-dimensional. We compute the distance matrix $d_{iv,jw}$ from the correlation matrix $C_{iv,jw}$ as

$$d_{iv,jw} = \frac{1 - C_{iv,jw}}{2} .$$

This mapping causes strong correlations to result in small distance values, while anticorrelations lead to large distance values. This is desirable to fulfill our goal of encoding correlations as distances among colors even though, in other cases, different mappings might be preferable, see Chapter 10.

The resulting distance matrix can be used for applying a distance-preserving 3D embedding. Multi-dimensional scaling (MDS) [373] minimizes the stress function and, thus, optimizes for preserving the distances. We, therefore, choose MDS for a 3D embedding. As a dimensionality reduction introduces error if the intrinsic dimensionality is larger than the dimensionality of the embedding, this method might introduce projection artifacts. However, the loss of information can be estimated by investigating the eigenvalues of the computation. Additionally, the similarity images provide an overview that could be accompanied by a more detailed analysis, as presented in the following chapters.



(a) Color coding.

(b) Similarity image.

Figure 8.2: The color coding is defined based on the coordinates in the embedding (a). The colors are then mapped back to the original spatial domain (b).

8.3.3 *Color Mapping*

The embedding can be used to map each of the embedded points $\mathbf{p}_{i\nu}$ that represents spatial sample $\mathbf{x}_{i\nu}$ of field ν to a color. We choose the CIE L*A*b* color space as Euclidean distances in the color space correspond to perceived distances between the colors and, thus, are perceptually uniform.

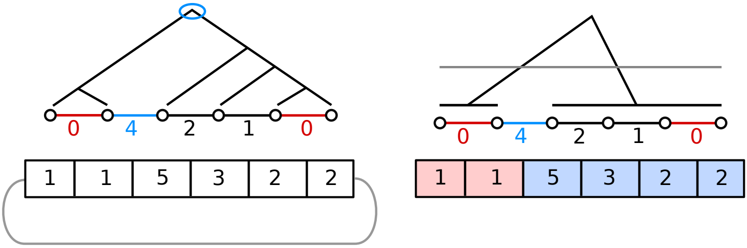
To use the range available in the color space as optimal as possible, we apply a rotation and a scaling to the points $\mathbf{p}_{i\nu}$. For the rotation, we use a heuristic, for which we place the most distant points on the diagonal of the cube. Therefore, we rotate the points around a rotation axis that is perpendicular to the diagonal and the direction vector from one of the most distant points to the other. The normalized rotation axis $\mathbf{r}$ can be computed as

$$\mathbf{r} = \frac{\mathbf{a}_{\max}}{\sqrt{3}\,\|\mathbf{a}_{\max}\|} \times (1,1,1)^T ,$$

where $\mathbf{a}_{\max}$ denotes the vector pointing from one of the most distant points to the other one. The corresponding rotation is given as [335]:

$$\mathbf{p}_{i\nu,\mathrm{rot}} = \begin{bmatrix} r_x^2 g(\alpha) + \cos\alpha & r_x r_y g(\alpha) - r_z \sin\alpha & r_x r_z g(\alpha) + r_y \sin\alpha \\ r_y r_x g(\alpha) + r_z \sin\alpha & r_y^2 g(\alpha) + \cos\alpha & r_y r_z g(\alpha) - r_y \sin\alpha \\ r_z r_x g(\alpha) - r_y \sin\alpha & r_z r_y g(\alpha) + r_x \sin\alpha & r_z^2 g(\alpha) + \cos\alpha \end{bmatrix} \mathbf{p}_{i\nu}$$

where r_x, r_y, and r_z denote the x, y, and z component of the normalized rotation axis and α is the angle between $\mathbf{a}_{\max}$ and the diagonal $(1,1,1)^T$ of the cube, $g(\alpha) = 1 - \cos\alpha$ and the diagonal $(1,1,1)^T$ of the cube.



(a) Binary partition tree.

(b) Watershed hierarchy.

Figure 8.3: The watershed segmentation is created based on an edge-weighted graph. A circular layout could be easily included by adding additional edges (gray). Edge weights can be computed as Euclidean distances where minima (red) and watershed edges (blue) are color-coded. The pruning line (gray) represents the cut in the watershed hierarchy (b), leading to a watershed segmentation.

The rotated points are first rescaled with the same scaling factor such that they lie in the cube $[0, 100]^3$. Afterwards, we translate all points such that they lie in the range $[0, 100] \times [-50, 50] \times [-50, 50]$. These ranges lie in the CIE L*a*b* color space, even though they do not present the full ranges of the color space. However, we chose these ranges as they allow us to preserve the relative distances and still use a wide range of colors, as shown for a color-coded embedding in Figure 8.2a. Then, the colors are applied to the image of the spatial region as shown in Figure 8.2b. The resulting image is referred to as a similarity image. Small perceived distances between the colors indicate a high correlation between the spatial positions, while large perceived distances indicate anticorrelations. Note that not all colors of the CIE L*a*b* color space can be shown in the RGB color space, which might introduce errors when viewing the images on the screen. However, following the same argument as for the embedding errors, these relatively small errors are acceptable for the overview visualization.

8.4 HIERARCHICAL SEGMENTATION

Regions of homogeneous colors in the similarity image represent regions of highly correlated temporal behavior. Therefore, applying a segmentation algorithm allows us to obtain regions of similar behavior that can be used to compute meaningful averages for a follow-up analysis which, then, would only depend on the internal variability of the data instead of on the input resolution of the dataset.

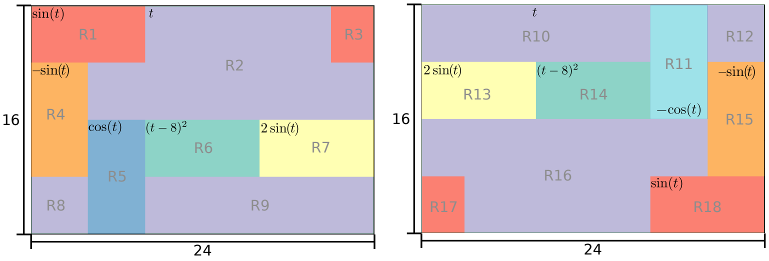
For larger flexibility in analysis approaches for the segmentation, we want to apply a hierarchical segmentation which will support the visual analysis on different levels of detail. Therefore, we choose a watershed algorithm that creates a hierarchy with homogeneous segments on the lowest level of detail as described in Section 8.1.1. A graph-based algorithm also supports different geometries, which allows us to segment, for example, the Earth's climate data on a spherical domain. The algorithm presented in Section 8.1.1 provides a quasi-linear runtime complexity in the number of pixels, and an efficient implementation is available [257].

To apply the algorithm, we define an undirected graph $G = (V, E)$ with vertices V and edges E from the image as shown schematically in Figure 8.3a. Each pixel is represented by one vertex, and edges connect adjacent vertices. To consider more complex geometries, we can insert additional edges. For example, to model the Earth's spherical surface, additional vertices between the pixels on the left and the right side of the image are inserted.

In the next step, the gradient between two vertices u_1 and u_2 is computed and assigned as a weight to the edge $e = \{u_1, u_2\} \in E$. We propose two methods for computing the gradient: Using the Euclidean distance between two color values leads to sharper edges. While this gradient computation is suitable for images with clear edges, it might lead to oversegmentation for smooth images. The Sobel filter [317] is a simple yet standard method for edge detection. This method is based on a gradient image, where each pixel stores the gradient magnitude. The weight for edge $e = \{u_1, u_2\}$ is computed as the mean of the gradient magnitudes assigned to the pixels corresponding to u_1 and u_2.

We use the similarity image in CIE L*a*b* color space to compute the segmentation. However, as the 3D embedding was mapped onto CIE L*a*b* colors by a linear transformation, the gradients of the color image correspond to those of the original embedding. Note that, in general, a computation of the segmentation is also possible by using higher-dimensional points, but we opted for a segmentation based on the 3D embeddings for easier interpretability. In this way, the segmentations can be directly compared to the similarity images.

After creating the graph that represents the image, we can apply the graph-based watershed algorithm leading to a watershed hierarchy. The watershed basins can be treated as the lowest level of the hierarchy (see Figure 8.3b). Thus, they can be either understood as leaves of the tree representing the hierarchical segmentation (see Chapter 9) or as a segmentation forming homogenous segments and, thus, a kind of superpixels for following analysis steps (see Chapter 10).



(a) Field v.

(b) Field w.

Figure 8.4: The artificial dataset contains two fields with a similar structure but different spatial positions of the individual segments. The function that was used for creating the time series is shown in each segment where equal functions are also encoded by color.

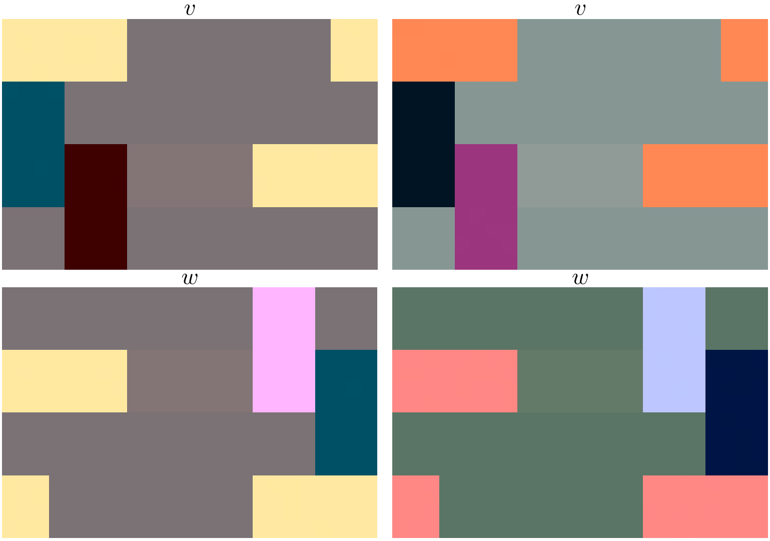
8.5 RESULTS

In the following, we will discuss some results for creating similarity images and the corresponding segmentations used in the following two chapters. First, we define a synthetic dataset to verify our approach and provide a more intuitive understanding of the meaning. Then, we will apply our approach to simulation ensemble modeling climate change.

8.5.1 *Synthetic Dataset*

For creating a synthetic dataset, we divide the spatial domain into different regions and create a time series for each spatial sample as shown in Figure 8.4. Thus, we obtain two fields, where the field v contains the same regions as the field w but in different spatial positions as we mirror the domain in both axes. We also switch the signs between R5 and R11. For each time series, we create 300 time steps with a temporal resolution of 0.1. We generate 10 ensemble members by adding random noise with a uniform distribution between $[0, 0.1]$.

Figure 8.5a shows the similarity images computed for both fields together. Here, the first three principal directions of the MDS cover 90.6% of the variability of the data. We can see that the regions R2, R8, R9, R10, R12 and R16 (see Figure 8.4) show the same color. The color of R6 and R14 is similar but not identical, as is to be expected for a relatively high correlation. Additionally, the segments R1, R3, R7, R13, R17, and R18 share the same color as the time series are, by definition, perfectly correlated. Segments R4 and R15 are defined to be anticorrelated



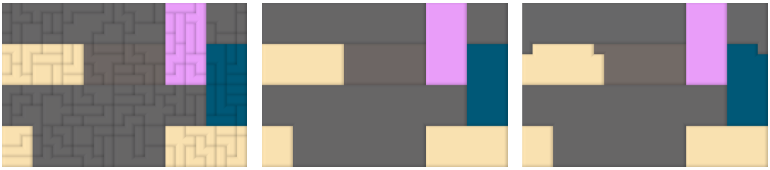
(a) Combined computation.

(b) Separate computation.

Figure 8.5: Computing the MDS for several fields together allows the interpretation that includes correlations between the different fields (a). If the similarity images are computed separately, distances between colors among the different images cannot be interpreted (b).

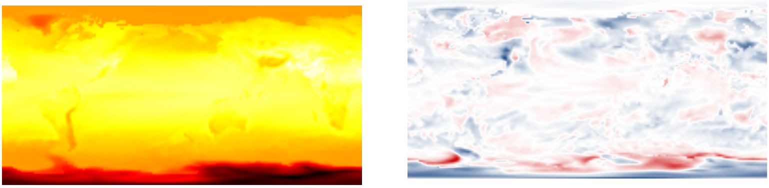
to these segments and, as expected, are perceived as a visually very distant color. Figure 8.5b shows the similarity images if both fields are treated separately and reveals that the colors between the images cannot be compared because the correlations among the different fields were not considered, which leads to a different embedding of the point cloud into the color space.

Comparing the two methods of computing multi-field similarity images, one can see that only the embedding containing both fields allows for deriving a conclusion about interactions between the fields. However, the computation for single fields is significantly faster as this computation only scales linearly with the number of fields compared to the quadratic scaling of the combined approach. Therefore, the preferable option for treating multi-field data strongly depends on the use case. If only a segmentation of the data is needed (see Chapter 10), the comparability between the color coding in both fields does not add additional value, which is why in this case, treating each field separately would be preferable.



(a) Watershed level 0.
(b) Watershed level 30.
(c) Sobel gradient filter.

Figure 8.6: At the highest level of detail (a), we observe an oversegmentation. At a lower level of detail (higher watershed level), using the Euclidean distance leads to sharp edges (b), which is not the case when using the Sobel filter (c).



(a) Temperature
(b) Temperature anomaly

Figure 8.7: Simulation data for the analysis of climate change. b) The heatmap shows the global spatial variation for temperature. b) When computing the temperature anomaly, the seasonal changes are removed.

Based on the similarity images, we can then compute a segmentation of the spatial domain as shown in Figure 8.6. Note that we use the visual encoding presented in Section 3.5.3. The segmentation of the field v when using the Euclidean distance for computing edge weights and a watershed level of 30 can be seen in Figure 8.6b. The segmentation resembles the domain boundaries. When considering the lowest level of detail, as shown in Figure 8.6a we see an oversegmentation that is caused by the noise added to the dataset and was to be expected.

We also compare the segmentation result using the Sobel filter for gradient computation. The result for a watershed level of 15 is shown in Figure 8.6c. Compared to the segmentation based on Euclidean distances, the boundaries between the segments are less accurate. From this observation, we can conclude that Euclidean distances yield better results for sharp segment boundaries.

8.5.2 *Global Climate Simulation*

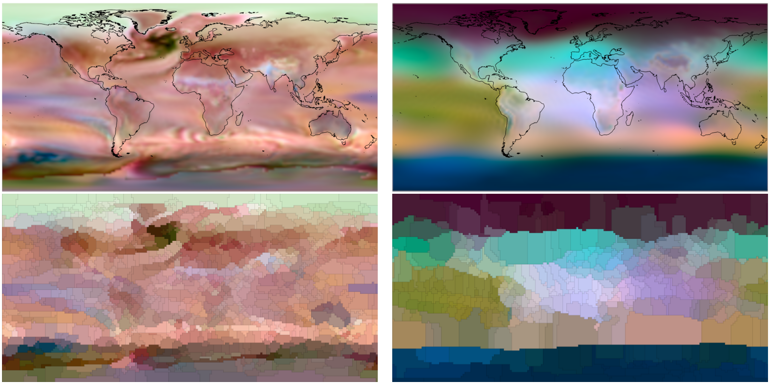
To show the practical applicability of similarity images, we apply our algorithm to simulations of climate change. We analyze the Max Planck Institute Grand Ensemble (MPI-GE) [223]. The simulation ensemble contains several scenarios based on different assumptions about greenhouse gas emissions. We will focus on scenario RCP8.5, simulated over the years 2006-2099. The ensemble contains 100 spatio-temporal ensemble members with a spatial resolution of 192×96. In contrast to the previous application areas, the dataset does not depend on a set of parameters. Instead, the initial conditions of the individual ensemble members are varied to cover the uncertainty in climate predictions. The dataset contains multiple fields, including the surface air temperature, near-surface air temperature, sea level pressure, and precipitation. One example of a single timestep for the sea surface temperature is visualized in Figure 8.7a.

Especially the temperature fields are dominated mainly by the seasonal cycle. Therefore, we derived the respective fields' anomalies by subtracting the respective fields' mean monthly values based on historical data. This procedure is also referred to as studying the anomalies with respect to the climatological mean. Analyzing the anomalies allows for investigating climate variability and identifying features of interest to domain scientists like the El Niño phenomenon. The anomaly for January 2006 in the sea surface temperature is shown in Figure 8.7b.

We compute the similarity images for the MPI-GE dataset. To reduce computational costs, we compute all similarity images individually. A suitable approximation algorithm like Landmark MDS [268] or Pivot MDS [48] could be used for computing joint similarity images as the distance matrix would be very large.

At first, we study the mean monthly surface temperature (ts) and obtain the image shown in Figure 8.2b. Here, we see a significant difference in the colors of the northern hemisphere and the southern hemisphere which can be explained by the seasonal differences. Additionally, one can clearly identify the continents. In the northern hemisphere, they are shown in pink, while the oceanic regions are shown in yellow, and in the southern hemisphere, the land mass is shown in blue while the ocean is significantly darker.

However, the strong seasonal fluctuations might hide more interesting features in the climate data. Therefore, we also consider the anomalies for each field. The similarity images for the surface temperature anomaly (ts_anomaly) are shown in Figure 8.8a. One clearly visible feature is the dark green region in the North Atlantic. The unique color points towards



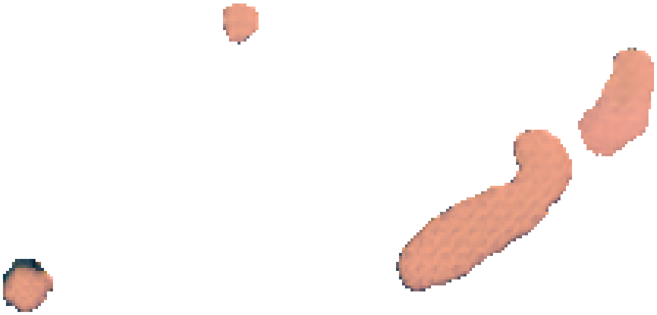
(a) tsAnomaly. (b) pslAnomaly.

Figure 8.8: The similarity image of the two anomaly fields reveals different spatial scales of variation. They show climate anomalies like the North Atlantic Worming Hole that remain hidden when analyzing the full data instead of the anomaly. For better orientation, the coastlines are overlayed.

a climate anomaly. Indeed, this region is well known to climate scientists as the so-called North Atlantic Warming Hole. The characteristic region for the El Niño phenomenon is encoded in a light violet. Additionally, we observe various Antarctic regions that stand out.

As a second example, we investigate the pressure anomaly. The similarity image shown in Figure 8.8b reveals variations on a larger scale. Additionally, the borders of the continents are significantly less pronounced, and only some of them are visible at all. For this field, the Arctic and Antarctic region stand out. Between those two regions, one only observes few different colors which form large connected regions.

The different granularities of the variations also become visible when observing the corresponding segmentations. Both segmentations shown in Figure 8.8 were created using the lowest watershed level. Due to the smaller regional variations, the segments for the pressure anomaly are significantly larger than those for the temperature anomaly. Working on the level of segments as superpixels instead of the individual data points of the original data thus reduces the data's complexity. Depending on the field, we can reduce the data from 18.432 spatial samples in the original resolution to 432 (sea level pressure) to 1075 (precipitation anomaly) segments. This results in a data reduction to 2.34% for the sea level pressure and 5.83% for the precipitation anomaly. The values for all fields can be found in Table A.2 in the appendix.



(a) Slice 75.

(b) Slice 100.

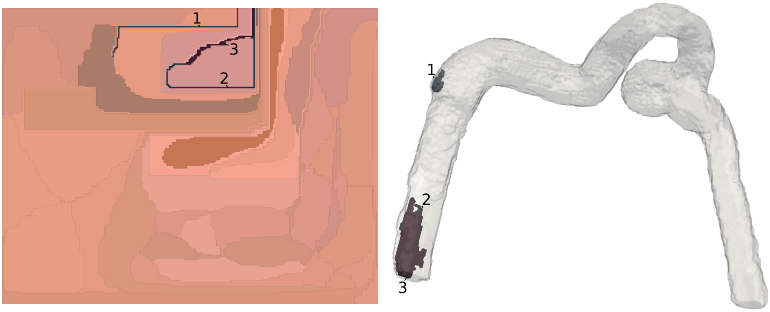
Figure 8.9: Two slices of the similarity volumes for the 3D blood flow dataset reveal high correlations among the spatial positions inside the vessel, encoded as a similar pink color.

Additionally, the analysis complexity depends on the internal structure of the data instead of the input resolution, which might allow for a significant reduction of computational costs without losing too much information. The similarity images and corresponding segmentations of the remaining fields and their anomalies are presented in Appendix A.2.

When observing the variation of the data covered by the three dimensions used in the similarity images, we see a wide range between the different fields. While for the surface temperature as shown in Figure 8.2b, 85.4% of the variation is covered, the pressure anomaly similarity image conveys 36.5% and the surface temperature anomaly only 26.7%. The percentages for the remaining fields are presented in Table A.1 in the appendix. However, increasing the target dimensionality for the MDS is not possible as color spaces are three-dimensional. Therefore, we propose to use the similarity image as an overview visualization and closely link it with other visualizations that allow for identifying projection artifacts, as we will explain in the following chapters.

8.5.3 *3D Blood Flow Ensemble*

To demonstrate the broader applicability of our approach and show a proof of concept for a 3D dataset, we apply it to the blood flow dynamics dataset that we also used in Section 4.8.1. Here, we consider that most of the 3D spatial domain does not belong to the vessel and, thus, does



(a) Embedding.

(b) Rendering.

Figure 8.10: Only three segments show a distinct color from the others (a). The 3D visualization (b) reveals that two of these segments are located at the inlet, while the third one is located at the vessel wall.

not contain data of interest for the correlation analysis. Therefore, we apply a mask to leave out these regions, and for the following analysis, only include the voxels inside of the vessels leading to $76,278$ spatial samples included in the analysis. This also shows that we can apply the approaches in the analysis workflow on data domains of various shapes.

We remove three runs from the dataset containing only 4 timesteps to create a comparable time domain. We use the first 11 time steps of all the other runs with a duration of 0.25 s each. In the analysis of the temporal variation of the ensemble members (see Section 4.8.1), it becomes clear that most variation appears in the earlier time steps. The similarity image, which could also be considered a similarity volume due to the 3D spatial domain, is created using PMDS, where we use 1% of the voxels as sample points.

Some slices of the volume are shown in Figure 8.9. We see that most regions are correlated, showing a similar pink color. Variations are mainly visible at the borders of the vessel. As the slices only present a subset of the data, we also compute the segmentation of the spatial domain. For reducing the number of segments, we choose a watershed level of 150. The resulting 3D segmentation can be embedded in 2D using the algorithm presented in Chapter 3. For coloring the segments, we choose the mean color of the corresponding voxels in multi-dimensional space. The embedding is shown in Figure 8.10a. We see only three segments that deviate significantly by showing a very dark color, indicating that the temporal evolution in these regions is not correlated to the other regions. Two regions share a joint boundary while the third one is separated. To better understand the spatial positions, these three regions are shown in a rendering in Figure 8.10b. One region (marked as 1) occurs at the vessel wall after the vessel diameter slightly extends. The two other regions (marked as 2 and 3) are located at the vessel's inlet. Here, the inlet velocity profile is fixed, which might allow for less variation over time.

8.6 DISCUSSION

In this chapter, we presented similarity images to provide an overview of spatio-temporal ensemble data. Perceived distances between colors in the images encode correlations between the different spatial regions. The image can be used as a starting point for an in-depth analysis of the simulation ensemble. We propose to apply a watershed segmentation. On its lowest level of detail, the segmented images provide highly correlated regions that reduce the complexity for future analysis. By computing a watershed hierarchy, segmentations on different levels of detail can be determined. These watershed hierarchies allow for an interactive investigation of the data on different levels of detail. Precomputing the segmentation supports analysis at an interactive rate which we will discuss in detail in the following chapters.

Note that similarity images can also be computed for datasets of higher dimensions leading to (multi-dimensional) similarity volumes. However, these volumes are more complex to visualize but could be shown by using, for example, slice-based visualizations. The bottleneck when computing similarity images is the 3D embedding. The underlying distance matrix scales quadratically with the number of data points, which grows quickly, especially for 3D datasets or datasets with even higher dimensions. This challenge can be addressed using MDS approximation techniques for large data, like Pivot MDS, as we did for the 3D blood flow data.

While the similarity images provide an overview of correlations between the different spatial regions and allow for spotting interesting anomalies, dimensionality reductions might induce projection artifacts. When observing the percentage of the variability in the data that the first three dimensions can cover, we find that we lose a significant amount of information for some fields. Therefore, similarity images do not suffice for a comprehensive analysis but should be supported by additional visualizations to avoid misinterpretations due to projection artifacts.

9

UNCERTAINTY-AWARE HIERARCHICAL CORRELATION ANALYSIS

Similarity images, as presented in the previous chapter, allow for an overview of correlations in a given ensemble. However, they might contain projection artifacts introduced by the dimensionality reduction to three dimensions. Additionally, they neither contain more detailed information about the uncertainty nor potential time lags between the segments nor allow for investigating the temporal behavior directly. We propose an interactive visual analysis approach for an uncertainty-aware hierarchical correlation analysis to address these challenges. We use hierarchical watershed segmentation, which allows an analysis on different levels of detail. We also include the uncertainty introduced by the ensemble structure and time lags between the time series in the analysis.

After providing an overview of the analysis tasks and our analysis approach in Section 9.1, we discuss how to compute correlations on multiple levels of detail in Section 9.2 and then detail our visual design in Section 9.3. Section 9.4 will present results from the climate simulations domain.

The results in this chapter are based on

> **M. Evers***, K. Huesmann* and L. Linsen, Uncertainty-aware Visualization of Regional Time Series Correlation in Spatio-temporal Ensembles, *Computer Graphics Forum* 40, No. 3: 519-530 (2021) (* The authors contributed equally.)

As mentioned in the previous chapter, all authors contributed to discussing ideas, writing, and editing the manuscript. The front end, including the visualizations, was mainly implemented by Karim Huesmann, while I implemented most of the back end, including precomputations. The results were created by both of us.

9.1 OVERVIEW

For the analysis in this chapter, we use the similarity images and a hierarchical correlation as described in Chapter 8 as a starting point. Using this data, we can analyze the spatio-temporal simulation ensemble with respect to the following analysis tasks:

T9.1 Analyzing the *correlations on multiple levels of detail*. It should be possible to refine the level of detail locally to cover regions of interest in more detail without adding additional complexity to other regions.

T9.2 Incorporating potential *time lags*. Time lags between time series might indicate causalities.

T9.3 Visualizing the *uncertainty*. The uncertainty originating in the ensemble structure should also be visually conveyed. Thus, we want to show the variability over the simulation runs.

T9.4 Visualizing the *time series* explicitly. Direct visualization of the time series data of selected samples supports understanding the temporal evolution.

Based on these tasks, we create an interactive visual analysis system. As a preprocessing step, we compute multi-level correlations that allow for adapting the level of detail at interactive rates (see Section 9.2). To achieve sufficiently fast computation times, we base the following steps purely on the segments and their hierarchy. Multiple linked views support the analysis of different facets of the data. A region visualization (see Section 9.3.1) provides spatial context to the correlation heatmap (Section 9.3.2). An uncertainty-aware time series visualization (see Section 9.3.3) provides direct visualization of the underlying temporal evolution. All views are closely linked by coordinated interactions (see Section 9.3.4).

9.2 MULTI-LEVEL CORRELATIONS

We precompute a multi-level correlation hierarchy for varying the level of detail on interactive rates (task T9.1). For this step, we start with a hierarchical watershed segmentation described in Section 8.1.1. The segmentation result can be stored in a tree data structure as shown in Figure 9.1. Further data can easily enrich this data structure.

Each tree node encodes one segment, while edges connect the segment to the corresponding parent and child segments. The leaves of the tree

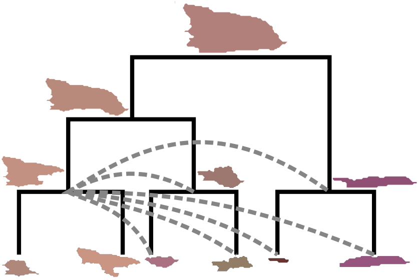


Figure 9.1: The hierarchical segmentation can be stored as a tree where each node corresponds to a segment. Multi-level correlations are computed among segments belonging to the tree's different branches.

are formed by the segments at the lowest level of the hierarchy. As these segments, in the following, are the smallest unit that we work on, the remainder of the approach does not scale with the resolution of the original data but instead with the information contained in the data. Thus, when increasing the resolution without varying the data itself, the size of this tree stays constant. In general, it significantly reduces the amount of data, see Section 8.5.2.

We store aggregated data in our tree data structure to obtain interactive rates. For each segment, we compute the mean of the time series over each spatial sample. The means are computed for each ensemble member separately. The mean time series are stored in the corresponding nodes of the tree and can be accessed for in detail-analysis (task T9.4). On the lowest level of detail, we can assume that the sample points are sufficiently similar such that the mean of the time series is representative. However, this is not necessarily the case for higher levels of detail anymore. To estimate the error introduced by considering means, especially of larger and more heterogeneous segments, we store the minimal and maximal pairwise correlation of its sample points for each segment. This allows for identifying segments with a high internal variation and enables the users to find projection artifacts that might have led to a misleading color coding in the similarity image (see also Section 8.5.2). We use the linearity of the mean for fast computation of the time series means. Only the means of the leaves need to be computed directly from the original data, while the means for the segments represented as internal nodes can be computed as area-weighted means of the mean time series stored in their children's nodes.

To store the spatial information, we assign each segment's spatial sample points to the tree's leaves. For determining cutting levels, we store the

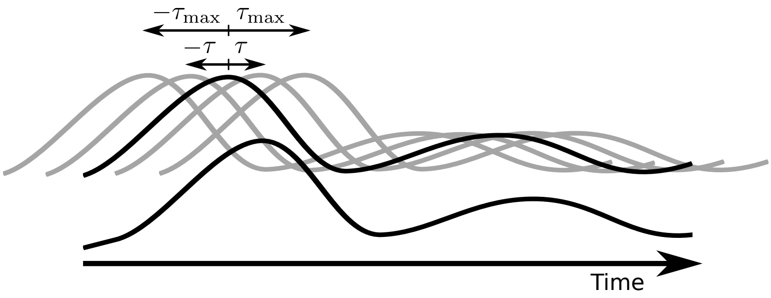


Figure 9.2: Computing correlations for different time lags τ between the time series allows for finding relations that might hint toward causality in the data.

altitude of the watershed segmentation as additional information. This also allows for refining single segments for a more detailed analysis of selected spatial regions (task T9.1).

For investigating the correlations on different levels of detail, it is also necessary to compute correlations among segments of different levels of detail. Here, we compute pairwise correlations for each ensemble member. However, to save computational costs as well as reduce the data that needs to be stored, we limit the computations to pairs of segments that do not belong to paths of the tree and use the symmetry of the correlation matrix.

Besides finding regions of synchronized behavior, including a time lag in the analysis is of interest. A time lag corresponds to a temporal offset between two time series, as shown schematically in Figure 9.2. Time-lagged correlations among regions hint towards causality among the corresponding temporal evolutions. The time-shifted cross-correlation $\rho_{ij,k}(\tau)$ for ensemble member r_k, regions i and j and time lag τ can be computed as

$$\rho_{ij,k}(\tau) = \frac{E[(\bar{s}_{ik}(t) - \bar{\mu}_i)(\bar{s}_{jk}(t+\tau) - \bar{\mu}_j)]}{\bar{\sigma}_i \bar{\sigma}_j} ,$$

where $\bar{s}_{ik}(t)$ and $\bar{s}_{jk}(t)$ are the mean time series of segment i and j, $\bar{\mu}_i$ and $\bar{\mu}_j$ are their means, and $\bar{\sigma}_i$ and $\bar{\sigma}_j$ are their standard deviations. $E[\cdot]$ describes the expected value. Note that in this chapter, we perform a single field analysis and, thus, do not include the index for the field. Based on a maximum time lag $\tau_{\max}$, we compute the correlations for all $\tau \in \{-\tau_{\max}, \dots, \tau_{\max}\}$. As the maximum time lag is very application-specific, we leave the choice of an accurate value to the users.

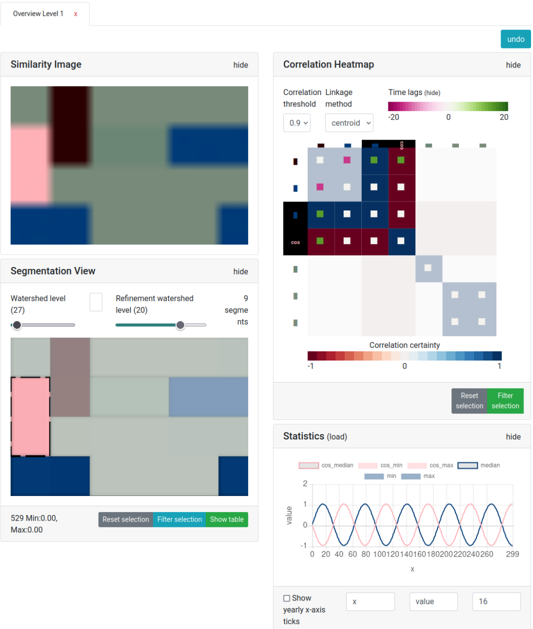
When computing all correlations for a pair of segments, this results in $(2\tau_{\max}+1)m$ correlation values for m ensemble members for each pair of segments. Depending on the choice of $\tau_{\max}$, this might lead to large amounts of data. To reduce this number, we consider only the strongest correlation value of the different time lags for each ensemble member obtaining only m correlation values. For the strongest correlation, we consider both positive and negative correlations. Thus, we use the maximum absolute value to determine the strongest correlation and store the sign to differentiate between strong correlations and strong anticorrelations.

For a global, regional ensemble analysis, the exact value of the correlation is less important than the existence of strong correlations or anticorrelations. Therefore, we threshold the correlation values while keeping the sign. Thus, we obtain the values -1 for strong negative correlations, 0 for weak correlations, and 1 for strong positive correlations leading to a vector of ternary values for each pair of regions. By computing the average over all ensemble members, we obtain a signed probability of whether the correlation exceeds the threshold, which covers the uncertainty within the ensemble. In the case of a positive value, it provides the percentage of ensemble members whose correlation exceeds the threshold. In the case of a negative value, the absolute value provides the percentage of runs with correlations below the negative threshold. As we assume that the ensemble members are relatively similar, it is improbable that strong negative and positive correlations occur in the same pair of segments. However, if it does occur, we alert the users for further investigation.

The choice for an optimal threshold value is unclear before the analysis and depends on the data. Therefore, we support multiple threshold values for which the multi-level correlations can be precomputed. The user can interactively switch between the different thresholds during the interactive analysis session. Precomputations for many thresholds require a large amount of memory, but for most applications, a relatively small number of thresholds should suffice. Note that up to this point, all calculations are precomputations carried out before the analysis sessions and allow for interactive analysis.

9.3 VISUAL DESIGN

In the following, we will discuss the design decisions to address the analysis tasks and explain the coordinated interactions. A screenshot of the resulting application is shown in Figure 9.3.



Correlation threshold 0.9

Linkage method centroid

Watershed level (27)

Refinement watershed level (20)

9 segments

529 Min:0.00, Max:0.00

x | value | 16

Figure 9.3: The visual analysis tool allows for studying correlations among different spatial regions while including time lags between the time series.

9.3.1 *Region Visualization*

While more abstract visualizations are helpful for deeper insights into the data, an explicit visualization of the spatial information is important. Especially in fields like climate research, geographical information is a key component in interpreting insights about the data. The visualization of 2D domains can be achieved easily by map-like visualizations. However, for 3D datasets, the spatial visualization of segmentations is more challenging because volume visualizations like direct volume rendering or visualizing the surfaces of the segments might lead to occlusion. Another option would be using slice viewers, but they only show a subset of the data at once and, thus, do not provide a complete overview of the dataset. When embedding the segmentation presented in Chapter 3, one obtains this overview but loses the spatial context. Therefore, we propose to use the map-like visualization of the embedding for an overview of the structure of the segmentation and link it to a surface rendering in which selected segments can be shown in 3D to avoid occlusion.

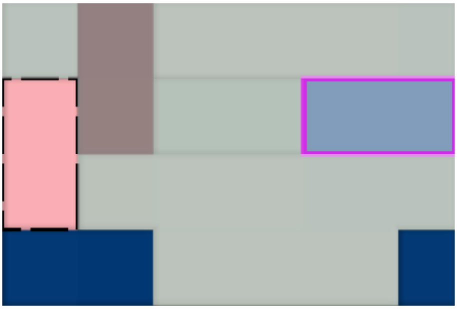


Figure 9.4: The region visualization shows the different segments. Selected segments are shown in full saturation, while the others are shown as desaturated. Annotated segments are shown with a black dashed boundary. For linking with coordinated views, segments that are hovered over are highlighted by a pink boundary.

For 2D spatial data, we show segments created based on the user-defined watershed level in a 2D map. Each segment is colored by the mean color of the corresponding sample points of the similarity images to provide the correlation information of the original similarity image also in the segmentation visualization. Showing the mean color also reflects the choice to work on the means of the spatial regions for the following analysis steps. However, this color coding might lead to similar colors of neighboring segments.

Different alternatives are available to ensure the separation of the segments. White boundaries between the segments, as used in the original approach [99], provide clearly visible separations between the different segments but also require additional screen space. While this visualization provides a simple layout that can be easily enriched by the additional encoding of the segment boundaries, it is not immediately applicable to visualizing an embedding of 3D segmentations (see Chapter 3). For 2D embeddings of 3D segmentations, a boundary between non-adjacent segments may be necessary if they do not share a joint boundary in the 3D space. If white boundaries separated the spatial segments, a different visual encoding would be needed for the segment separators.

Another option would be shading, as discussed in Chapter 3, that clearly highlights boundaries between different segments. This visualization also directly generalizes to 3D embeddings as it was originally designed for this purpose. While both visual encodings have strengths and weaknesses, we use shading-based visualization to keep the visual encoding consistent for different dimensions. An example of the spatial region visualization is shown in Figure 9.4.

To facilitate the analysis and interpretation of the data, the users can interactively label segments, which are segments of particular interest. Then, the labels are used in all visualizations. To highlight these segments in the region visualization, we draw a black dashed border of the segment on top of its original visualization.

As we also show the similarity image and the segmentation visualization, the users can assess the segmentation quality. Further, it facilitates the choice of a suitable watershed level. When interactively changing the watershed level, the map visualization adapts to the different levels of detail.

The dataset can be analyzed by the visual information-seeking mantra "Overview first, zoom and filter, then details on demand" [313] because the continuous level-of-detail hierarchy allows for adaptively refining segments (task T9.1). Thus, we allow a more global analysis to reduce the visual load. The level of detail can be modified to select segments, allowing for a direct comparison of segments on different levels of detail. Interactive rates can be obtained because the correlations are precomputed.

9.3.2 *Correlation Heatmap*

The correlations among the different spatial regions should be investigated, including time lags (task T9.2) and the uncertainty caused by the ensemble (task T9.3). Visualizing correlations of spatial regions is closely related to climate networks which are commonly visualized by using a node-link diagram on top of a map. However, drawing many edges on top of a map leads to much visual clutter even without encoding additional information like time lags or uncertainty. Instead of using a node-link diagram that directly contains the spatial information, we opt for a heatmap that allows for including additional information without suffering from occlusion, as shown in Figure 9.5. The heatmap is closely linked to the region visualization described in Section 9.3.1 to avoid losing the spatial information.

Our visualization is based on a heatmap and shows the matrix of correlation probabilities whose computation is described in Section 9.2. Here, we cover the uncertainty (task T9.3) in the ensemble by showing the percentage of ensemble members that exceeds a certain correlation threshold. A heatmap scales well with the number of segments, supports enhancements with additional information, and (with a suitable ordering) allows for identifying structures of interest. The heatmap can be

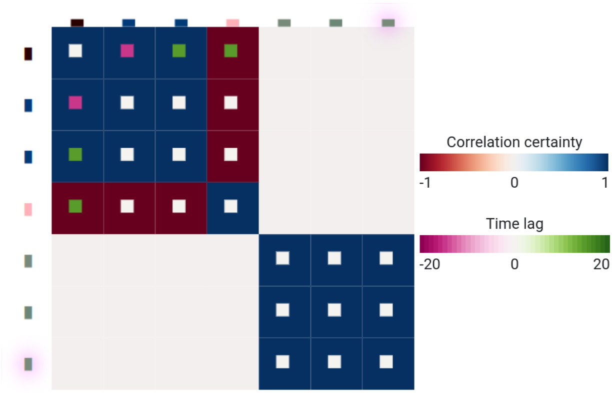


Figure 9.5: The correlation heatmap shows the percentage of ensemble members that exceed the threshold (here: 0.8). The small squares in the center of each heatmap cell encode the time lag. The color labels on the axes correspond to the segment colors. The pink highlighting of the labels in the lowest row and the rightmost column encode that the user hovers over the corresponding segment in the region visualization.

linked to the region visualization by applying consistent color coding. Any assigned labels are also linked between both views. Additionally, coordinated interactions (see Section 9.3.4) establish a stronger linking that also supports differentiating between segments of similar color.

In the heatmap, we use color to encode the correlation probability, which is the percentage of runs that surpasses the negative or positive threshold. A diverging color map allows for differentiating between positive and negative correlations, so we use a red-white-blue colormap where red encodes a high probability for negative correlations and blue a high probability for positive correlations. The heatmap allows different ways of filtering the data. For focusing on strong correlations among segments, the users can hide rows and columns that do not contain strong correlations to any other segments, as this reduces the size of the heatmap. Further size reductions can be obtained by filtering the segments included in the analysis, as discussed in more detail in Section 9.3.4.

The spatial regions that identify the rows and columns of the heatmap are not ordered. However, the ordering of the matrix significantly influences the interpretability of the result as a suitable ordering allows for identifying groups of highly correlated segments. The matrix of correlation probabilities can be interpreted as an adjacency matrix where each entry corresponds to the strength of the connection between the different segments. Thus, we can apply matrix reordering methods commonly used to analyze adjacency matrices in network analysis.

Behrisch et al. [29] derived a guideline for choosing reordering algorithms based on the application case. As we aim at cluster identification to identify groups of correlated segments, we follow their recommendation to choose hierarchical clustering for this purpose. In this chapter, we use the SciPy implementation for hierarchical clustering [356]. The outcome of hierarchical clustering strongly depends on the choice of the clustering algorithm. We leave the choice of the linkage algorithm to the user because it strongly depends on the data to analyze. We support the following algorithms:

- Single linkage (minimal distances between two points)
- Complete linkage (maximal distances between two points)
- Average linkage (unweighted pair-group method for arithmetic averages (UPGMA))
- Centroid linkage (based on the distance of centroids of the clusters)
- Median linkage (average of child centroids instead of calculating the centroid from the original points)
- Ward's minimum variance method

The user can interactively switch between the different linkage algorithms during the visual exploration of the correlation probability matrix.

In addition to investigating the correlations, it is interesting to see for what time lag τ the maximum correlation is observed. The time lag is included in the heatmap visualization by showing a color-coded square in each matrix cell where the color of the square represents the time lag. As the time lag τ can be positive or negative, we again opt for a diverging color map. However, as these colors are used together with the color map representing the signed probabilities, we choose a pink to green color map that is distinguishable from the red to blue color map used in the same visualization. The time lags can also be used for filtering. Only regions whose correlations exceed the threshold for the selected time lag are shown when selecting a single time lag.

9.3.3 *Uncertainty-aware Time Series Visualization*

To better understand the data, the actual time series should be visualized (task T9.4). We want to visualize the segment means for each ensemble

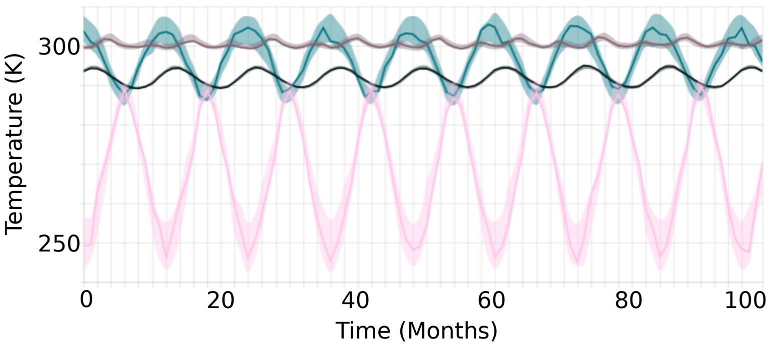


Figure 9.6: The uncertainty-aware time series visualization shows the ensemble median curve (as defined for functional boxplots) together with a band displaying the range covered by the ensemble. The curves are colored according to the corresponding segments.

member. To show the uncertainty of the data, we choose a representation inspired by functional boxplots [328] as shown in Figure 9.6. The median time series, as defined for functional boxplots, is the time series with the largest band depth [218] and corresponds to the most central time series of all ensemble members. Thus, the median represents an actual ensemble member that is part of the ensemble. In traditional functional boxplots, the median is surrounded by the 50% central region and the maximum envelope without outliers, where outliers are drawn explicitly. However, as we find the climate ensemble data to be rather noisy, this would lead to many outliers resulting in a cluttered visualization (see Section 2.3.5 for a synthetic example). Instead, we surround the median with bands that spread the entire range. Thus, it covers the whole variation over the ensemble members, so we also refer to them as 100% bands.

To visually link the time series visualization to the other views, we apply a consistent color coding and use the mean segment colors to encode the time series. Even though the colors of correlated time series might be similar, this does not impose problems. Instead, encoding the correlation by the perceived distances between the colors facilitates the analysis as the different colors for negatively correlated or uncorrelated time series directly show that there is no correlation.

9.3.4 *Coordinated Interactions*

The individual visualizations (see Sections 9.3.1 to 9.3.3) cover different facets of the ensemble data. For comprehensive data analysis, these views should be used together. We achieve this by closely linking the views. Besides the visual linking by using consistent color coding in all visualizations, we also link the views using coordinated interactions.

The region visualization provides spatial information for the correlations shown in the heatmap. Hovering in one of the visualizations highlights the corresponding segment in the other visualization to link the segments between both visualizations. The segments in the region visualization are highlighted by showing a pink frame around the segment as shown in Figure 9.4. When hovering over the segmentation view, the color label of the corresponding segment in the heatmap is highlighted by adding a pink shading as shown in Figure 9.5.

Selecting a set of regions to limit the analysis to a smaller amount of data is also possible. Spatial regions can be selected directly in region visualization or by brushing in the heatmap. In both views, the unselected segments are rendered in faint gray, which allows the users to see the boundaries between the segments for context while also putting the focus on the selected segments. The time series view shows the time series of the segments selected in one of the other views. It is also possible to select specific time lags or correlations. These selections can be made by clicking on the respective regions in the color map and are immediately applied to the heatmap view.

The data can be filtered based on a selection with either of the previously discussed methods. The analysis of the filtered data can be performed in a separate tab, allowing the users to switch back to the previous level of detail easily. After filtering the data, the same interactions as for the entire dataset are available. Thus, the data can be analyzed on various levels of detail while conveniently switching between them. In views showing filtered data, the segments corresponding to regions excluded from the analysis are grayed out in the region visualization. In the heatmap, this data is not shown to create additional space for showing the filtered data in more detail. The more detailed investigation also includes changing the watershed level for the segmentation of single segments or the whole filtered set of segments. A top-down analysis approach is supported by allowing filtering and refinement on different levels of detail. As each filtering result is shown in a separate tab, our visualization approach allows quickly returning to higher levels of detail realized by switching to the respective tab. Thus, this approach provides provenance information.

9.4 RESULTS

We will validate our approach by applying it to a synthetic dataset and comparing our findings to the ground truth. Then, we demonstrate the effectiveness by analyzing the 2D MPI-GE dataset and discuss the results with a domain expert. Finally, we show how our approach can be applied to a 3D blood flow simulation.

9.4.1 *Synthetic Dataset*

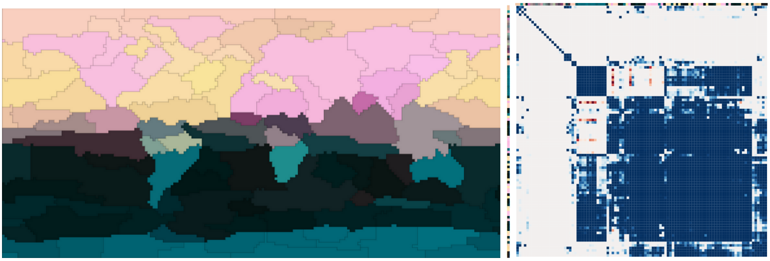
We use the dataset described in Section 8.5.1 to verify our approach. As we focus on analyzing single fields in this chapter, we only use the field v.

For the following analysis, we choose a watershed level of 30 to avoid oversegmentation but instead obtain one segment per region with unique behavior. Figure 9.5 shows the correlation heatmap for this data. The green segments, corresponding to R3/R8, R4, and R6 in Figure 8.4a, show a strong correlation that is larger than the threshold of 0.8. At the same time, these regions are not correlated to any other region, as was to be expected by the definition of the data. The correlation between regions with linear variation over time (R3/R8 and R4) vanishes if the threshold is increased to 0.95, which aligns with the expectations. The pink region (R2) is anticorrelated to the blue segments (R1 and R7), which follows the definition as $\sin(t)$, and the multiples thereof are anticorrelated to $-\sin(t)$. The dark brown segment (R5) is correlated to the blue segments where the time lag is ± 15 (depending on the segment), which corresponds to a shift of approximately $\pm\pi/2$ and agrees with the expected shift between $\sin(t)$ and $\cos(t)$. Overall, the observations follow the expectations based on the definition of the dataset.

9.4.2 *2D Climate Ensemble*

To show the applicability to real-world data, we apply our approach to the MPI-GE dataset (see Section 8.5.2). The results were discussed with Michael Böttinger from the German Climate Computing Center (DRKZ), who has many years of experience working with climate data. As we only focus on single fields in this chapter, we only focus on the mean monthly surface air temperature and its anomaly.

When computing the similarity images, one obtains the results discussed in Section 8.5.2. For the hierarchical analysis, we first consider the temperature and start with a watershed level of 20 that leads to 111 segments as shown in Figure 9.7a. When observing the correlation



(a) Segmentation. (b) Heatmap.

Figure 9.7: The segmentation at watershed level 20 (a) and the corresponding correlation heatmap with a threshold of 0.9 (b) for the temperature field of the MPI-GE exhibit various correlated regions.

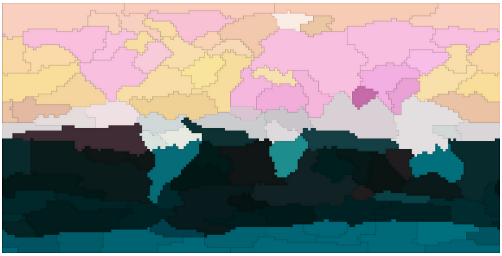
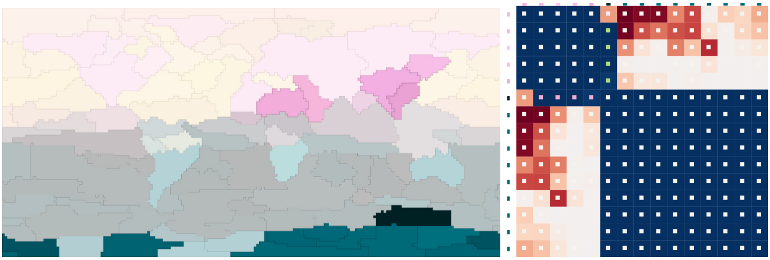


Figure 9.8: The regions are filtered for being correlated with a time lag of 6 months.

heatmap shown in Figure 9.7b, we observe several correlations among the different segments. Also, some anticorrelations (red shades) stand out. To investigate the time lag, we can use the time lag color map and select a time lag of 6 months corresponding to the seasonal shift between the northern and southern hemispheres. Thus, we select all segments that are correlated or anticorrelated with this time lag to any other segments. The segmentation shown in Figure 9.8 reveals that this applies to the majority of segments outside the equatorial region, which is very plausible.

Next, we want to investigate the negative correlations we identified in Figure 9.7. Therefore, we filter the regions for those that show a negative correlation to at least one other region resulting in the segmentation and correlation heatmap shown in Figure 9.9. We immediately see that the uncertainty of showing a correlation < -0.9 varies because the shades of red vary. Some regions in East Asia and North Africa are strongly an-



(a) Segmentation.

(b) Heatmap.

Figure 9.9: The segments shown in Figure 9.7 are filtered for negative correlations.

ticorrelated with regions in Antarctica. The pink segments are positively correlated to the northernmost Antarctic segment (black) when applying a time lag of 5 months (pink/green encoding in the centers of the cells). From this observation, we can deduce that the positive correlation with a time lag of 5 months is stronger than the anticorrelation without a time lag.

As in the previous chapter, we also investigate the temperature anomaly to avoid the domination of the seasonal cycle. When observing the segmentation as shown in Figure 9.10 we can see that most segments share a similar color while few stand out. We start our analysis by selecting the green segment at the left and right border. Its time series is shown in Figure 9.11a. Besides the general increasing trend attributed to global warming, we observe a variation at a timescale of three to five years. This regular increase in temperature is related to the El Niño phenomenon, which is one of the most well-known climate anomalies and is characterized by temperature variations in this selected region.

The dark purple area in the center, corresponding to the North Atlantic Ocean, sparked the domain expert's interest and is, therefore, selected for further analysis. Figure 9.11b shows the corresponding variation over time. Here, the increase in temperature is significantly smaller than in other regions, and in some parts, it even seems to decrease. This phenomenon is known in climate research to occur in this region, also called a "warming hole". Probably, this behavior is linked to a change in ocean circulation [184]. Besides the limited increase in the overall temperature, we observe a substantial increase in the annual fluctuations, which becomes apparent in the prominent oscillation starting after 30 years.

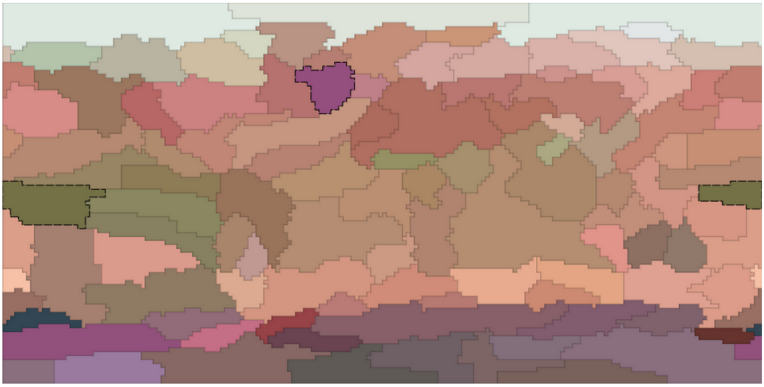


Figure 9.10: In the region visualization for the temperature anomaly (watershed level 20), two interesting regions are marked by a dashed border. The green segment is characteristic of the El Niño phenomenon, while the violet region represents the North Atlantic Warming Hole.

9.5 DISCUSSION

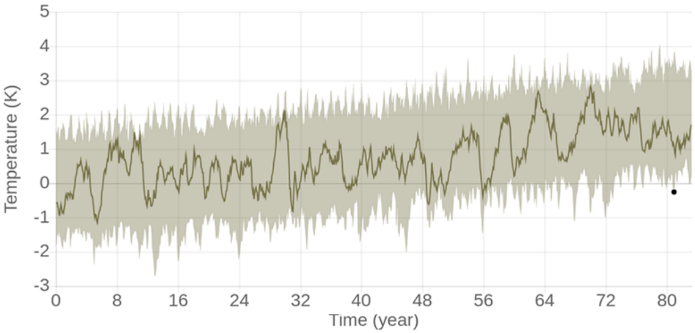
We presented an interactive visual analysis approach[1] for studying hierarchical correlations in spatio-temporal ensemble data. Based on a hierarchical segmentation of the similarity image, correlations in the data can be analyzed, including the time lag and the uncertainty introduced by the ensemble structure of the data. Consistent color coding and interactions closely link the coordinated views. The approach was verified based on synthetic data and proved helpful in investigating correlations in climate ensembles.

The domain expert rated our tool as helpful in finding climate phenomena. He linked some patterns immediately to well-known climate anomalies but also saw the potential to investigate new patterns and phenomena. He pointed out that including typical climate research attributes would be helpful for a more detailed analysis.

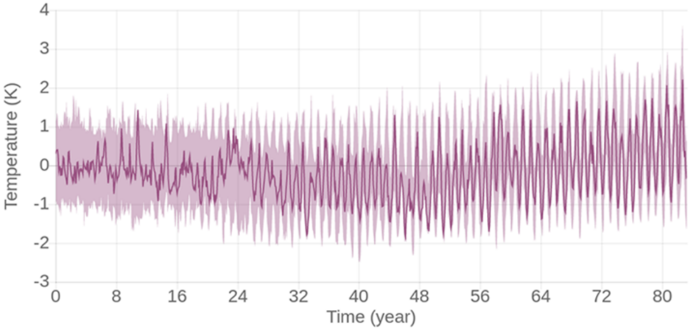
The approach only depends on two parameters which are the maximum time lag and the correlation threshold. As both parameters are very application specific, we leave their choice to the users. However, domain knowledge makes these parameters simple to define, and no parameter fine-tuning is required.

---

[1] `https://github.com/marinaevers/regional-correlations`



(a) El Niño.



(b) North Atlantic Warming Hole.

Figure 9.11: The time series visualization shows the temporal evolution and its ensemble variation of the segments marked in Figure 9.10.

The analysis presented in this chapter scales quadratically with the number of segments because we compute pairwise correlations. However, if the number of segments is very large, but a high level of detail is not required, it is also possible to prune the lowest levels of the hierarchical segmentation and only perform the analysis on a lower level of detail. The number of segments in the datasets we used was relatively small, allowing for analysis at interactive rates without further simplifications.

In this chapter, we only work on single field data. However, simulation ensembles like the MPI-GE dataset commonly contain multiple fields. The correlation heatmap and the time series visualizations can be easily generalized for a multi-field analysis. While the spatial information can be visualized with several juxtaposed region visualizations, the limited screen space limits the scaling of the analysis. Therefore, in the following chapter, we will present an alternative workflow that emphasizes the correlation analysis among multiple fields.

10

INTERACTIVE CORRELATION ANALYSIS IN MULTI-FIELD CLIMATE ENSEMBLES

The individual analysis of single fields of simulation ensembles, as presented in Chapter 9, allows for investigating various phenomena. However, a comprehensive analysis of multi-field ensemble data requires the joint analysis of multiple fields. The different fields can influence each other and are often closely related. For example, the El Niño phenomenon is a well-known climate anomaly that can be identified by increased sea surface temperature in the eastern and central Pacific Ocean during the winter. However, this temperature anomaly is also known to influence several other climate variables, like the precipitation in various regions of the world. Such interactions can only be investigated when considering multiple fields together.

In this chapter, we propose a visual analysis approach for analyzing the correlation among spatial regions between different fields of ensemble data. Our interactive visual approach supports the users in identifying regions of correlated temporal behavior without the need to choose a reference point. An embedding is created by using the correlations among spatial regions as distances. Users can then interactively select clusters that represent groups of spatial regions with correlated temporal behavior. The properties of the underlying time series can be investigated in more detail, where we include a geographical map for spatial context, a heatmap showing the correlations and the spread over the ensemble, a direct visualization of the time series data, and a spectral analysis. The utility of this approach is shown by analyzing the MPI-GE dataset with different fields.

We will first describe our workflow in Section 10.1 and discuss the necessary precomputations in Section 10.2. Then, we discuss our design choices for the different visual encodings (see Section 10.3). We verify our approach by applying it to a synthetic dataset with known ground truth (see Section 10.4) and show its usefulness by presenting two application use cases from the field of climate ensemble analysis 10.5.

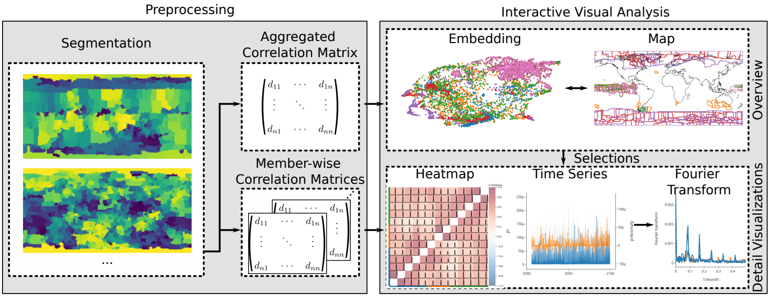


Figure 10.1: The analysis process builds on segmenting the similarity images used to create correlation matrices. In an interactive visual analysis, these correlations can be investigated on different levels of detail.

The results of this chapter are published as

> **M. Evers**, M. Böttinger and L. Linsen, Interactive Visual Analysis of Regional Time Series Correlation in Multi-field Climate Ensembles, *Workshop on Visualisation in Environmental Sciences (EnvirVis)*, 69-76 (2023)

All authors contributed to discussing ideas, writing, and editing the manuscript. I implemented the approach and created the results with Michael Böttinger, who provided helpful feedback from the domain experts' perspective.

10.1 WORKFLOW

The visual analysis workflow for analyzing correlations between multiple fields in spatio-temporal simulation ensembles is shown in Figure 10.1. It starts with segmentations of similarity images as discussed in Chapter 8. Here, we use only the lowest level of detail to obtain homogeneous regions where the temporal evolution of the field is highly correlated but still reduce the amount of data included in the analysis. Based on these segmentations, we can compute correlation matrices as detailed in Section 10.2. A web-based visual analysis tool allows for investigating these correlation matrices at interactive rates.

The analysis process starts with a 2D embedding which is explained in more detail in Section 10.3.1. Each point in the embedding represents a segment in a single field, and clusters of points represent highly correlated segments. The embedding is linked to a map view to provide spatial information of the corresponding segments (see Section 10.3.2), where the outlines of segments that can be selected in the embedding view are drawn on top of a geographical map which provides spatial context. The correlations among the spatial segments are also explicitly shown in a heatmap visualization of the correlation matrix to study them in more detail, identify projection artifacts and study the variation over the ensemble. We discuss the visual design of the heatmap in more detail in Section 10.3.3. For analyzing the time series directly, we provide a similar visualization as discussed in Section 9.3.3. However, as we focus on multi-field correlation analysis in this chapter, we show the time series of multiple fields together, allowing for a better understanding of correlations between different fields. For this purpose, we show an individual y-axis for each field that allows for reading off the individual values. As the number of segments in the multi-field analysis is significantly higher than in the single-field analysis, a color coding based on the mean color of a similarity image would be overwhelming. Therefore, we change the color coding to represent the field which provides consistency in this analysis tool and allows for differentiating the various fields. For investigating periodicities in climate ensembles, we include a Fourier analysis as described in Section 10.3.4. All visualizations are linked by consistently using the fields for color coding. Further, linking is provided by brushing interactions that allow for differentiating segments in the same field, which would be assigned the same color.

10.2 PREPROCESSING

As a starting point for this analysis, we use the similarity images and correlations discussed in Chapter 8. For multi-field simulation ensembles, the similarity images can be computed jointly or separately. While a joint computation would provide related colors, the colors will not be used in the later stages of this approach. Therefore, we use separate computations of the similarity images to reduce the computational cost and increase the accuracy of the segmentation. A joint similarity image computation would lead to a more complex dataset as a significantly larger amount of data would be included. This would also lead to a larger loss of information because the target dimensionality remains 3D.

We use only the highest level of detail of the hierarchical watershed segmentation presented in Section 8.1.1. Thus, the resulting segmentation contains only segments of highly correlated data points, which we could assume to be homogeneous such that a meaningful aggregation of the time series is possible. We compute the mean time series of all spatial samples for each segment and ensemble member, leading to n time series per segment for n ensemble members. In the following steps of the analysis approach presented in this chapter, we will only work on the level of segments and their corresponding means. This dramatically reduces the data size (see Section 8.5.2). Similar to the approach presented in Chapter 9, the number of time series included in the analysis only depends on the complexity of the data but not on the spatial resolution.

We compute two correlation matrices to support correlation analysis on different levels of detail. For a joint analysis of multiple fields, we use the segments of all fields together to compute the correlation matrices. At first, we compute an aggregated correlation matrix that contains the pairwise correlations between segments for the whole ensemble. Here, we compute the correlations between the concatenated time series described in Equation 8.1. Note that no normalizations are necessary to include different fields because we are computing correlation values that analyze the trends instead of the exact values. The resulting correlation matrix will be mapped to a distance matrix, used for embedding, and shown directly in the heatmap visualization.

We also compute a member-wise correlation matrix to cover the variability over the ensemble. For this one, we compute the pairwise correlations between the segment means for each ensemble member individually, leading to n correlation values for each segment. This matrix will show the distribution of the correlation values in the heatmap.

10.3 VISUAL DESIGN

In this section, we will discuss the design decisions for the visual analysis tool. A screenshot of the tool's user interface is shown in Figure 10.2.

10.3.1 *UMAP Embedding*

We start the analysis process with a 2D embedding of the segments where each segment of each field is represented by a point. For computing an embedding as shown in Figure 10.2, we choose the uniform manifold approximation and projection (UMAP) [228]. UMAP preserves clusters, which is our primary goal. Compared to other standard techniques like t-distributed stochastic neighbor embedding (t-SNE) [348], UMAP tends to better preserve the global structure of the multi-dimensional data.

For computing the embedding, we use the aggregated correlation matrix computed based on the ensemble-aggregated time series, which contains a single correlation value for each segment. Analogous to the mapping for the similarity images described in Section 8.3, we map the correlation to distances in the range of $[0, 1]$ where 1 corresponds to highly

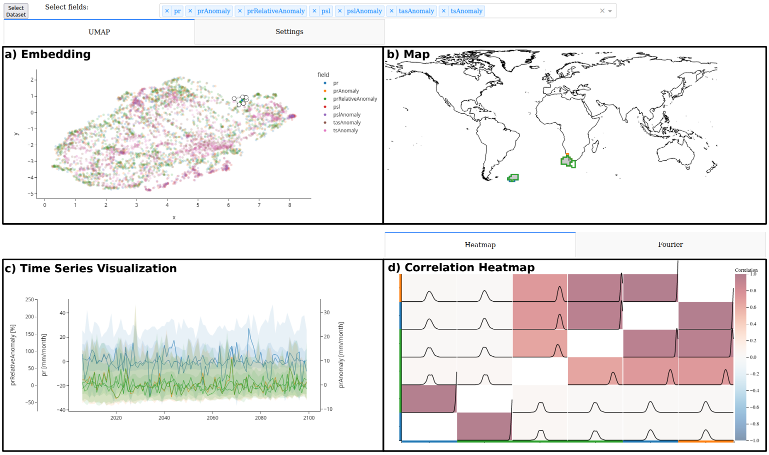


Figure 10.2: The visual analysis tool for multi-field simulation ensembles enables an analysis of correlations between regions of multiple fields. A UMAP embedding (a) of the segments allows for cluster identification and selections where selected segments are drawn on a map for spatial context (b). Selected time series can be shown as a graph over time (c), while the correlations and the distribution of correlations over the ensemble are shown in a heatmap (d).

correlated segments and 0 to anticorrelated segments. However, depending on the use case, showing strongly anticorrelated segments close together can be desirable as they are also strongly related. Therefore, we allow the user to switch to an alternative distance mapping, where strong anticorrelations are also mapped to 1, and uncorrelated segments (correlation of 0) are described by a distance of 0. For this option, the distance between segments i and j can be computed as

$$d_{iv,jw} = 1 - |C_{iv,jw}|, \tag{10.1}$$

where $C_{iv,jw}$ is the correlation between segment i of field v and segment j of field w.

Based on the resulting distance matrix, we compute an embedding containing the structure of the whole dataset, including various fields. To distinguish to which field the segment corresponding to an embedded point belongs, we color code the fields.

However, the resulting UMAP embedding depends on a density parameter and the number of neighbors considered for each point. As the optimal choices of these parameters are unclear, we allow the users to vary these parameters during the analysis process. Thus, it is also possible to find different features in the data by investigating embeddings created for varying parameter settings. For the results presented in this thesis, we considered a minimum distance of 0.1 and 500 nearest neighbors for a more global overview.

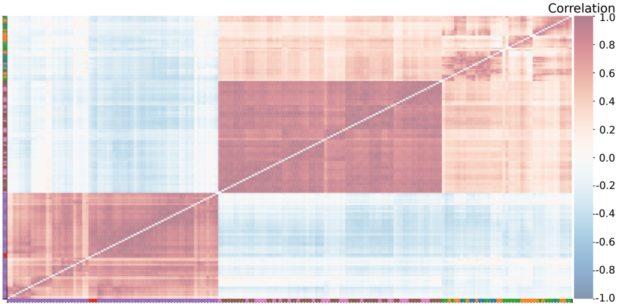
If many fields are investigated at once, the scatterplot visualization of the embedding can quickly become cluttered. Therefore, the users can hide specific fields in the embedding that are occluding interesting structures in other fields. Hiding the data points in the embedding does not affect the other visualizations. If a field should be generally excluded from the analysis, this can also be done globally, influencing all visualizations.

Brushing in the scatterplot allows for selecting groups of points. The segments corresponding to these points can be used in the other visualizations for a more in-depth analysis. The selected points are highlighted visually by decreasing the saturation of the other points. To facilitate the analysis process, it is also possible to interactively label groups of points. Thus, the user can add, for example, notes about the geographical position to the embedding.

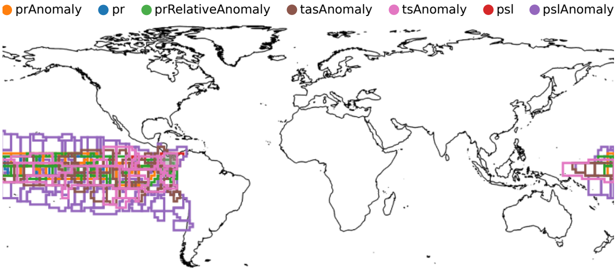
10.3.2 *Map View*

While the UMAP embedding provides an overview of the correlations among the different segments, it does not encode the spatial positions of the segments encoded by each point. Therefore, we link the embedding to a map view that provides the spatial information as shown in Figure 10.2b. Each segment corresponding to user-selected points is drawn as an outline colored according to the field. Using outlines instead of areas allows for drawing segments of several fields together, even if they overlap. Geographic context, for example, for analyzing climate data, can be provided by drawing the segments on top of a world map. Here, we choose to show the outlines of the continent. Depending on the dataset, the context can be easily exchanged. Updating the segments when changing the selection leads to a direct connection between the embedded points and the segment's spatial positions.

Investigating the correlations among segments in certain spatial regions is also of interest. For this purpose, the lasso selection in the map view allows defining a spatial region in which all segments should be selected. Here, we define a segment as belonging to the selected region if it at least partially overlaps with the region drawn by the lasso selection. When selecting a region in the map view, the corresponding points are marked as selected in the embedding. This allows for investigating the



(a) Heatmap.



(b) Map view.

Figure 10.3: The heatmap (a) shows correlations between the time series of selected segments. For spatial context, the segments are also shown in the map (b). The pressure fields (psl and pslAnomaly) are anticorrelated to the temperature fields (tsAnomaly and tasAnomaly) and the precipitation fields (pr, prAnomaly, and prRelativeAnomaly). Overall, the correlation between the different temperature fields is higher than between the segments of the other fields.

correlation structure in the selected region but also investigating it in the context of correlation to unselected points that are shown as desaturated.

10.3.3 *Heatmap*

A heatmap allows for visualizing the correlations directly based on a color coding as shown in Figure 10.3a. While the correlations are also encoded as distances in the embedding described in Section 10.3.1, a dimensionality reduction might introduce projection artifacts. The explicit encoding in a heatmap, thus, avoids misinterpretations while also investigating the correlation values in more detail. In the heatmap, we

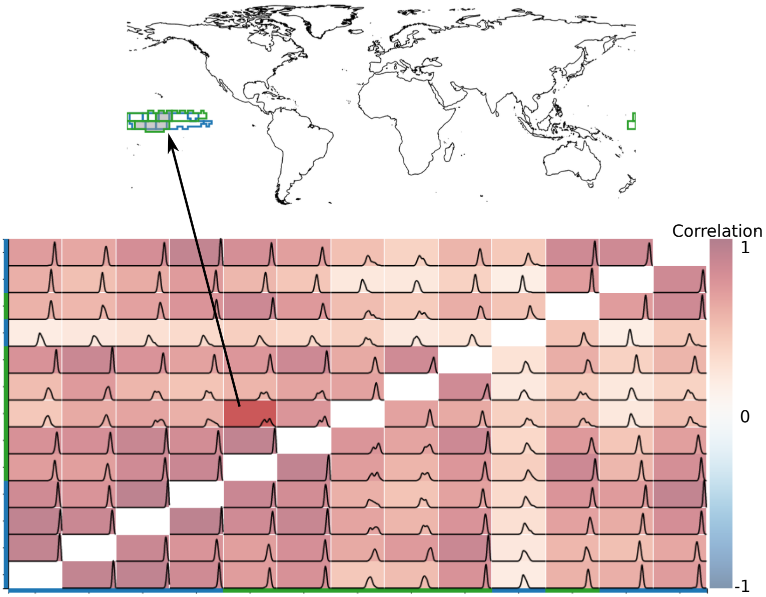


Figure 10.4: Hovering over the heatmap highlights the corresponding segments in the map view by filling the areas with a semi-transparent gray color.

color-code the aggregated correlation values, which cover the correlation over the whole ensemble. As a color map, we use a diverging blue-white-red color map such that blue encodes anticorrelations and red encodes correlations.

The ordering of the heatmap's rows and columns significantly influences the interpretability and the ability to analyze groups of correlated segments [29]. Thus, we apply a matrix reordering similar to Chapter 9.3.2. We also use hierarchical clustering. For the results presented in this work, we used Ward's minimum variance method [365], which we found to yield good results. In the resulting matrix, similar rows and columns are grouped closely together, which results in blocks of highly correlated or anticorrelated segments, as can be seen in Figure 10.3a. We label the rows and columns of the matrix by color-coding the fields. This links to other visualizations using the same color coding (see Figure 10.3b). Additionally, it scales well with the number of segments shown in the heatmap, and it allows for easily deducing the fields for which the time series are correlated or anticorrelated. To obtain more accurate information about the spatial locations, we link the heatmap again to the map view, analogous to Chapter 9. However, as we use outlines here to show the segments, we cannot use the outlines to encode which segment is hovered over. Instead, we use the areas, filling the corresponding segments with a semi-transparent gray color as depicted in Figure 10.4. The reduced opacity allows for keeping the underlying map and potential segments belonging to other fields visible.

The color coding of the correlation values covers aggregated correlations over the whole ensemble. To encode the spread of the correlation values over the ensemble members, we enrich the heatmap by showing the distribution of the correlation values. This encoding enables the users to determine whether the ensemble correlation value is representative and how much the correlations for the different ensemble members deviate. For this purpose, we use the second precomputed correlation matrix, which contains n correlation values per segment pair for n ensemble members. We apply a kernel density estimate for smoothing the distribution, where we use the Epanechnikow kernel [96], which minimizes the mean square error and is a standard approach for this purpose. The resulting distribution is shown as a graph overlay to the color coding for each matrix cell, see Figure 10.4.

If huge matrices are visualized, the distribution graphs might provide too much overlap and hide the color coding. To address this issue, we allow the users to interactively hide the distributions depending on their analysis goals and the number of segments included in the heatmap visualization. Another option would be to apply the approach for responsive matrix cells [159] which allows for showing detailed visualizations inside of matrix cells on demand. However, as we would either show the same encoding in all cells or none of them, responsive matrix cells would require too much interaction. Hiding the distributions, reordering the matrix, and using colors for labeling the rows and columns provides a good scaling with the number of segments, as seen for a larger matrix in Figure 10.3.

10.3.4 *Fourier Analysis*

Besides showing the time series in a line chart as described in Section 9.3.3, studying the frequency spectrum of the time series is also interesting. Especially in the field of climate research, many phenomena are periodic. Besides obviously periodic events like annual fluctuations, climate anomalies such as the El Niño/Southern Oscillation (ENSO) phenomenon reappear in certain intervals. Thus, the correlation between segments is often closely linked to the (quasi-)periodic behavior. However, these events are not always perfectly periodic, and the frequencies

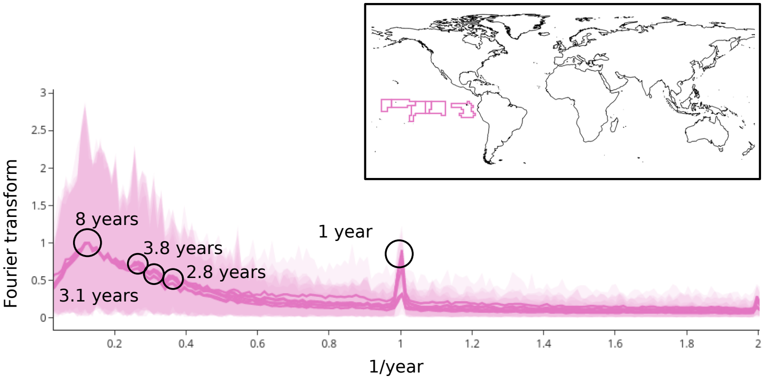


Figure 10.5: The Fourier spectra of the selected segments for the temperature anomaly shows several peaks of different size. While seasonal fluctuations cause the narrow peak at a frequency of 1/year, the peaks between 0.2/year and 0.4/year might be related to the El Niño phenomenon. Besides the peaks in the mean spectra, there is a substantial variation over the ensemble, as indicated by the bands.

might vary over time, for example, due to climate change. We include a Fourier analysis in our approach to study the frequencies and their variations. The users can select time-intervals in the time-series visualization for which the Fourier transform is computed. Thus, spectra for different time intervals can be compared. Jänicke et al. [166] propose to use wavelet power spectra for similar purposes. While their method facilitates the investigation of temporal evolution, it does not support showing the spectra of several regions in the same graph, which is essential for our goal of studying relations between different segments of potentially different fields.

We remove the linear trend before computing the Fourier transform to avoid global warming dominating the Fourier spectrum. Different options for a Fourier analysis of an ensemble of time series exist. The easiest method would be computing the Fourier spectrum of the mean time series. However, this could lead to losing many interesting features as different frequencies are averaged out. Instead, we compute the Fourier transform of each ensemble member individually and compute the mean Fourier spectrum to reduce noise. To still cover the variability of the ensemble, we surround the mean curve by a band ranging from the minimum to the maximum value as shown in Figure 10.5.

We normalize the Fourier spectra to the $[0, 1]$ range because we are only interested in the frequency. Additionally, the normalization facili-

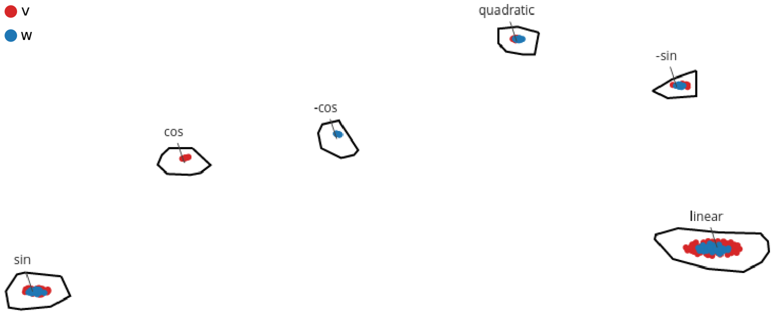


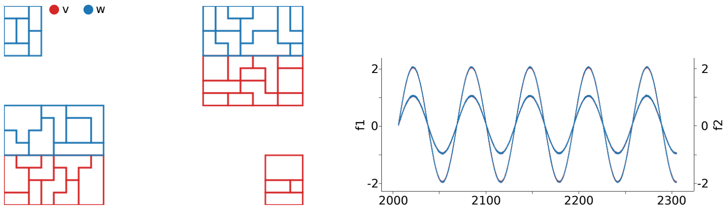
Figure 10.6: The embedding of the synthetic dataset shows distinct clusters for the different functions used. Here, the built-in annotation tool was used for adding labels to the different groups.

tates the comparative analysis of frequencies in Fourier spectra of different fields. The graph visualization shows the mean Fourier spectra with the corresponding ranges for all selected segments. For consistency, we color code the spectra according to the field that they belong to.

10.4 SYNTHETIC DATASET

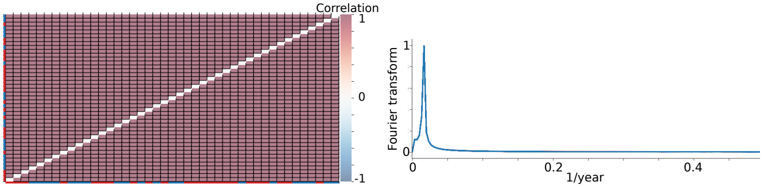
We verify our approach by analyzing the synthetic dataset presented in Section 8.5.1. We start with the UMAP embedding of the different segments shown in Figure 10.6. Here, we can see clusters that belong to the different functions used for creating the dataset. We can label the different groups for a better overview. Next, we investigate the cluster according to the two different but highly correlated sine functions in more detail. The analysis results are presented in Figure 10.7. When selecting the cluster, we observe that the spatial regions agree with our definition. Additionally, the time series visualization reveals that the selected spatial segments contain two different functions, while the heatmap shows that they are strongly correlated. When selecting a time interval and computing the Fourier transform, we obtain the result shown in Figure 10.7d. In this view, we cannot distinguish between segments corresponding to $\sin(t)$ and those corresponding to $2\sin(t)$. This behavior is caused by the normalization of the Fourier spectra. However, as we are only interested in the frequency spectrum, it correctly reveals identical frequencies in both functions. The spectra contain one very prominent peak at 0.017 per year, which agrees with the frequency used to create the data.

To further verify the heatmap visualization, we select a subset of points from the three clusters shown in Figure 10.6 in the right. Thus, we obtain



(a) Map view.

(b) Time series.



(c) Heatmap.

(d) Fourier.

Figure 10.7: The results for the synthetic dataset agree with the definition provided in Section 8.5.1. Here, the cluster labeled with sin in Figure 10.6 is selected. The map view allows for verifying that each cluster belongs to the correct segment (a). The time series visualization shows the two different sine functions (b), while the heatmap shows a strong correlation for the selected segments (c). Analyzing the Fourier spectrum of the full-time span (d) reveals a dominant peak corresponding to the sine frequency.

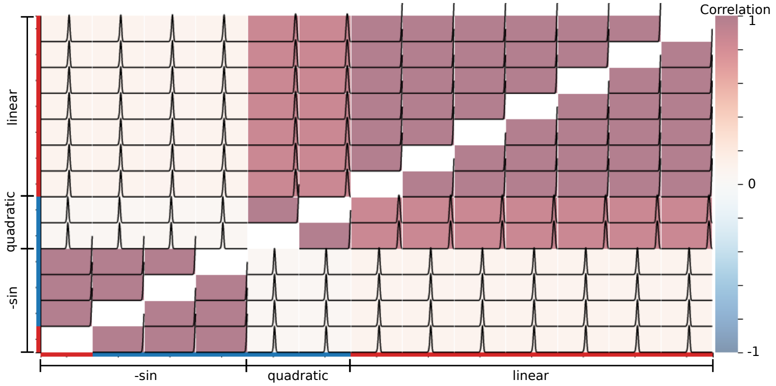
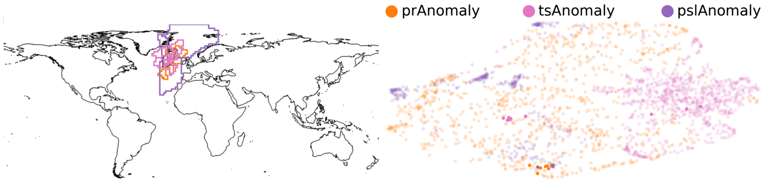


Figure 10.8: The heatmap shows strong correlations inside the clusters. While the linear and the quadratic segments are positively correlated, the periodic time series are neither correlated to the linear nor the quadratic ones.



(a) Map view. (b) Embedding.

Figure 10.9: When selecting the region around Iceland and the Azores (a), the points corresponding to the different segments are highlighted in embedding (b).

the heatmap shown in Figure 10.8, where we added labels according to the clusters. As expected, the correlations inside the clusters are very high. We observe a high positive correlation between the segments with a quadratic mean time series and a linear one, while we see no correlation between these segments and those with a sine as a mean time series. When observing the distributions of the correlations of the ensemble members, we observe very narrow distributions, which agree with the definition of the dataset. As the ensemble members only vary by a small noise, we expect little variation in the correlation values.

10.5 CLIMATE ENSEMBLE ANALYSIS

To show the usability of our approach, we analyze the MPI-GE dataset described in Section 8.5.2. We present two use cases where we analyze well-known climate anomalies in our data, and after that, we discuss the feedback provided by the domain expert.

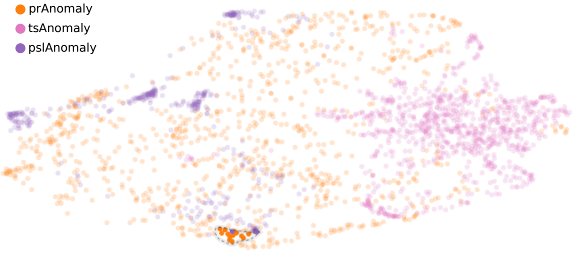
10.5.1 *North-Atlantic Oscillation*

The North-Atlantic Oscillation (NAO) is one of the dominating climate patterns in the Northern Hemisphere [162]. The oscillation can be characterized by fluctuations of the normalized pressure difference between the Icelandic Low and the Azores High, which is most prominent in winter. As the pressure difference influences westerly winds and storm tracks in the North Atlantic, the NAO directly influences many aspects of life in Europe, including agricultural harvest and energy supply and demand. While the pattern itself has been known for a very long time, there a still some unsolved questions, for example, regarding the predictability of the oscillation [162].

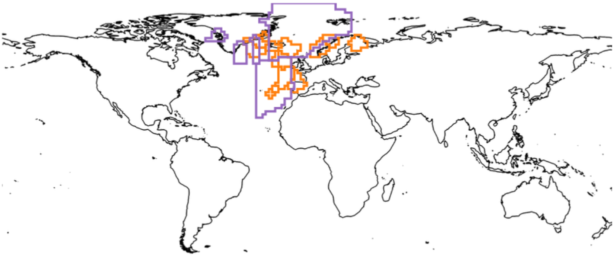
As the NAO is characterized by high air pressure in the Azores region and low pressure in Iceland, but we also want to investigate the relations to other fields, we include the pressure anomaly (pslAnomaly), the surface temperature anomaly (tsAnomaly), and the precipitation anomaly (prAnomaly). As we expect an anticorrelation between the pressure over the Azores and Iceland, we compute the UMAP projection using the absolute value of the correlations as described in Equation 10.1. Thus, we expect points representing highly correlated as well as anticorrelated segments to be located close together.

To analyze this phenomenon, we start by selecting the spatial regions around the Icelandic Low and the Acores High in the map view as shown in Figure 10.9a. Based on this, we can identify the locations of the corresponding segments in the embedding (see Figure 10.9b). Considering the highlighted points from the map selections, we select a set of points belonging to the pressure anomaly (pslAnomaly) and precipitation anomaly (prAnomaly) fields in this region, as shown in Figure 10.10a. In the map view, we verify that the selected segments for the pressure and precipitation anomalies are located around the Azores region and Iceland, while we also selected some additional segments. The heatmap in Figure 10.10c reveals a strong anticorrelation between the Azores High and the Icelandic Low. The distributions shown in the heatmap cells reveal little variation over the ensemble. When considering the precipitation anomaly fields, we can identify a correlation between the pressure anomaly in the region of the Icelandic Low to the precipitation anomaly in Southern Europe, while it is anticorrelated to the precipitation anomaly in Northern Europe. For the pressure anomaly in the region of the Azores High, we instead observe a strong anticorrelation to the precipitation anomaly in Southern Europe and a weaker, positive correlation to the precipitation anomaly in Northern Europe. Thus, our analysis approach does not only support finding the anticorrelation in the pressure as usually used as an indicator of the NAO but also studies its influence on other fields.

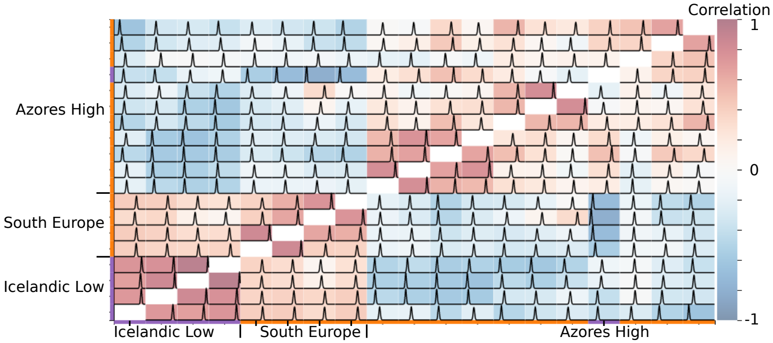
For analyzing the time series frequencies, we select four segments to reduce the clutter in the Fourier view. The resulting Fourier transforms are shown in Figure 10.11. In both fields, we see two dominant peaks. The first one, at a frequency of $1/\text{year}$, indicates the seasonal cycle. As we removed the seasonal variations based on historical data, these clear yearly variations indicate that the strength of the seasonal changes increased compared to the historical data. The second peak indicates repetitions twice a year, which can also be attributed to seasonal fluctuations.



(a) Embedding.



(b) Map view.



(c) Heatmap.

Figure 10.10: We select a set of points belonging to the pressure and precipitation anomaly (a). The segments belonging to these points are located northwest of Europe (b). The heatmap reveals correlations among the Icelandic Low and the Azores High and correlations and anti-correlations to the precipitation anomaly over Europe (c).

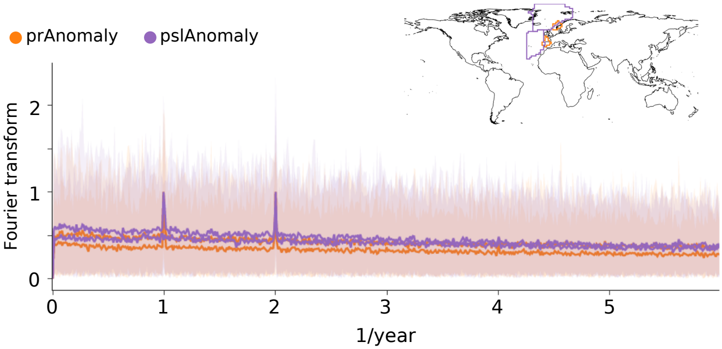
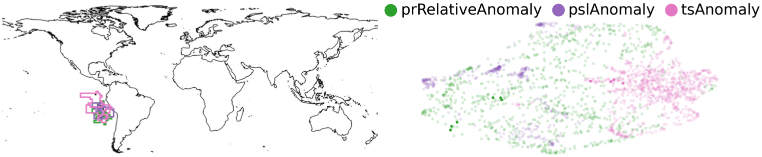


Figure 10.11: The Fourier transform of the segments shown on the map inlay reveals dominant peaks at frequencies of $1/\text{year}$ and $2/\text{year}$ for the pressure and precipitation anomalies. Besides these clear observations in the ensemble means, we see a wide spread of the ensemble.



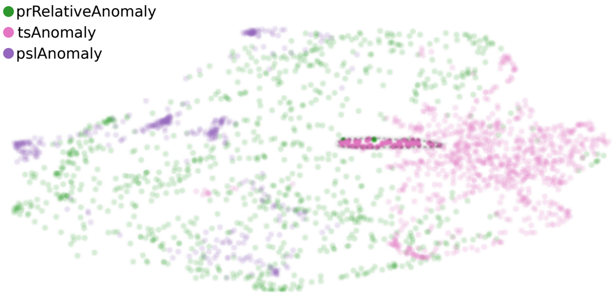
(a) Map view.

(b) Embedding.

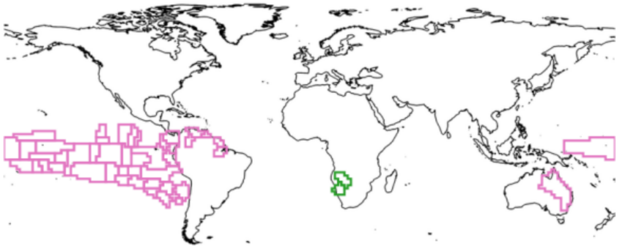
Figure 10.12: To investigate the El Niño phenomenon, we select the characteristic region in the map (a). The corresponding segments are shown with full saturation in the embedding (b), which allows for a selection to find correlated segments.

10.5.2 *El Niño/Southern Oscillation*

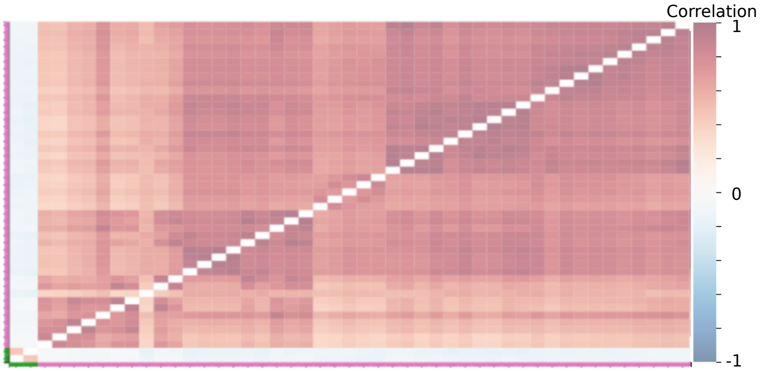
Next, we aim to analyze the El Niño phenomenon and its long-range interactions. Similar to the previous use case, we use Equation 10.1 to place correlated and anticorrelated points close together in the embedding. We start with a selection in the map view where we select a region west of South America as shown in Figure 10.12a. The positions of the points representing the selected segments are shown in Figure 10.12b. As the El Niño phenomenon is characterized by an anomaly in the temperature, we select a set of points around the temperature anomaly values and limit the selected fields. The selection is shown in Figure 10.13. In the map, we can see that the corresponding segments exhibit long-range correlations and anticorrelations. In the context of climate research, these



(a) Embedding.



(b) Map view.



(c) Heatmap.

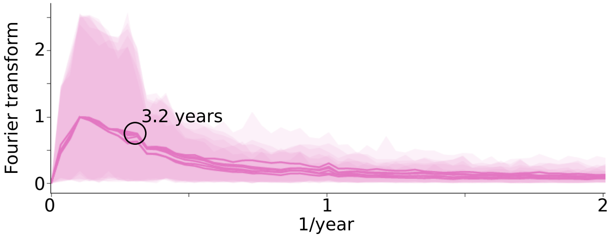
Figure 10.13: The selection of points in the center of the embedding (a) mostly belongs to the El Niño region west of South America (b), but some segments are also located in eastern Australia and Southern Africa. The heatmap (c) shows that the temperature anomaly segments are positively correlated with each other but negatively correlated to the precipitation anomaly.

long-range relationships are also referred to as teleconnections. When observing the map, we see that most segments are located west of South America. This region is known to be closely linked to the El Niño/Southern Oscillation (ENSO) phenomenon, which is the strongest interannual climate variability [202]. This phenomenon can be identified by a positive sea surface temperature anomaly during winter in the central and east Pacific. This anomaly occurs every few years and strongly influences the weather worldwide.

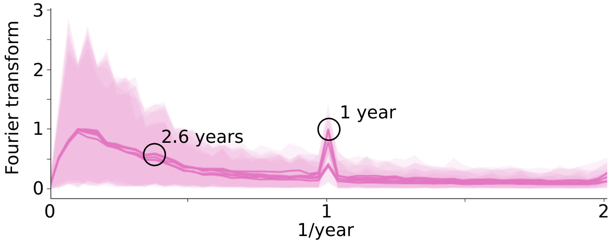
In Figure 10.13b, we can see that some of the selected temperature anomaly segments are located in eastern Australia or the north of South America. The segments belonging to the precipitation anomaly field are in Southern Africa. The heatmap (see Figure 10.13c) reveals a positive correlation among the different segments belonging to the temperature anomaly. We also observe an anticorrelation to precipitation anomaly in Southern Africa. Even though the anticorrelation is relatively weak, it is considered relevant as we observe different fields. According to the domain expert, the correlations among different fields are generally lower. These correlations are in agreement with the IPCC report [61].

As El Niño occurs in approximately regular intervals, it is of interest to investigate the Fourier transform. The spectrum for the whole time span is shown in Figure 10.5. The spectrum is dominated by a peak with a frequency of $0.12/\text{year}$. As this corresponds to a period of 8 years, it is not the common period for the El Niño phenomenon. We cannot yet explain the exact reason for this dominating peak. Another prominent peak occurs at a frequency of $1/\text{year}$. This peak corresponds to seasonal fluctuations. Several smaller peaks occur for period lengths typical for the El Niño phenomenon.

To study variations over time, we first investigate the Fourier spectrum for the years 2006-2035, shown in Figure 10.14a. Besides the dominant peak for small frequencies, we see a smaller peak corresponding to a period length of 3.2 years which can be related to El Niño. When observing the Fourier spectrum for 2070-2100, we immediately notice that the peak at a frequency of $1/year$ becomes significantly more dominant. This can be explained by increased seasonal changes related to global warming. Additionally, we observe that the small peak now corresponds to a period length of 2.6 years. Relating this peak to the one with a period length of 3.2 years for the earlier time interval indicates a shortening of the period length for the El Niño phenomenon. This behavior has been described before [202] and is considered linked to climate change. However, this peak is not very dominant, and in general, we observe a large spread of the ensemble variation around the ensemble means. Therefore,



(a) 2006-2035



(b) 2070-2100

Figure 10.14: Comparing the Fourier transform for different time intervals allows for identifying variations in the frequency spectra over time. For the temperature anomaly in the El Niño region, we identify peaks that indicate a shortening of the periods from 3.2 years (a) to 2.6 years (b). The strong peak at a frequency of 1/year indicates a change in the seasonal variations.

it is unclear if this observation is present in all ensemble members and requires further investigation.

10.5.3 *Domain Expert Feedback*

All the previously discussed analysis use cases were created in close collaboration and discussion with Michael Böttinger from German Climate Computing Center (DKRZ). The tool was developed in close collaboration with regular feedback meetings. After the first version of the tool was provided, he worked with the tool independently to generate the results presented in the previous subsections. This also led to several further improvements, including spectral analysis and annotations.

During the analysis process, he highlighted the possibility of investigating the data interactively. In the analysis process, the UMAP projection helped investigate correlated regions which can be compared to well-known climate phenomena. The additional option to treat anticorrelations as small distances further facilitated the analysis process when related data, in general, should be investigated.

Despite reproducing existing phenomena, the visual analysis approach also provided insights into the data that still needs to be fully explained. One remaining open question is the appearance of the prominent, broad peak at a frequency of approximately $1/8$ years in the Fourier spectrum of the El Niño region. Additionally, the importance of the reproducibility of the results for domain scientists was highlighted. While all results presented in this chapter are reproducible by selecting the same sample points, it would be significantly facilitated by saving the selections and allowing for loading single or even multiple of them.

10.6 DISCUSSION AND CONCLUSION

This chapter presents a new approach for studying correlations in spatio-temporal multi-field ensemble data[1]. Starting from an embedding, clusters of correlated or anticorrelated spatial segments can be identified. The correlations can be investigated in more detail in linked visualizations, including the spread over the ensemble. A direct time series visualization and the Fourier transform provide further insides into the temporal evolution.

Besides verifying our approach using synthetic data, we proved its usefulness by analyzing the MPI-GE. Besides identifying well-known climate phenomena, we also found features, like the 8-year period in the El Niño region, that we could not yet explain. A more detailed analysis of this phenomenon and future research on improving the computation and visual representation of uncertainty in Fourier transforms could yield further insights.

The approach presented in this chapter scales with the number of segments instead of the input dimension of the data, which leads to dependency only on the variability of the data. A high number of segments slows down the UMAP computation. However, as the embedding does not need to be recalculated regularly, this should not significantly influence the analysis process. The number of segments depends on the spatial variability and the number of fields included in the analysis process. Even though a higher number of fields leads to a more cluttered scatterplot, the scalability with the number of fields can be improved by choosing the most important fields directly in the visual analysis tool. Thus, our approach also works on subsets of data with many segments and fields where the fields can be changed during the analysis process.

---

[1] `https://github.com/marinaevers/multifield-correlations`

Our approach generalizes to other domains, especially in scientific simulations, where correlations between spatial regions are of interest. While we only discussed 2D examples in this chapter, all steps except the map view are directly applicable to 3D data as well. For providing spatial context, a suitable volume visualization like a volume rendering or a surface visualization of the different segments is suitable even though these visualizations might be cluttered for many fields. A generalization of the segmentation embedding presented in Chapter 3 to multi-field data would provide an additional overview of the spatial domain.

In this chapter, we focussed on analyzing correlations between several fields but did not include the time lags as in Chapter 9. A combination of the approach presented in this chapter with the analysis from the previous chapter would allow for a more comprehensive analysis.

Part IV

CONCLUSION

11

CONCLUSION AND FUTURE WORK

In this thesis, we proposed novel approaches for the interactive visual analysis of spatio-temporal simulation ensembles. While the analysis of ensemble data contains many facets and analysis tasks, we focus on analyzing parameter spaces and global correlations between spatial regions. For both topics, we presented approaches that allow for investigating simulation ensembles on different levels of detail. We will summarize our contributions before discussing future research directions and open challenges.

In the first part, we developed methods for investigating the parameter space and linking it to the simulation outcome. We presented novel approaches for finding and analyzing partitions of the parameter space, determining dependencies of the simulation outcome on the input parameters, and analyzing the sensitivity to the individual parameters.

At first, we proposed a general method to embed multi-dimensional segmentations or partitionings into 2D where the area and the boundary sizes are preserved (see Chapter 3). The embedding is created by describing the segmentation of the multi-dimensional space in a graph data structure which can be embedded in 2D using graph drawing algorithms. A cellular automaton optimizes the visualization to preserve the areas and boundary sizes. Our segmentation embedding allows for a comprehensive visualization of the segmentation. The practical applicability is demonstrated not only in 3D segmentations but also in the visualization of multi-dimensional parameter-space partitionings (see Chapter 4).

We presented an approach for a comprehensive analysis of multi-dimensional parameter spaces in Chapter 4. Based on semi-automatic clusterings in the similarity space of the simulation ensemble, we partitioned the parameter space to identify regions of similar simulation outcomes. For a distortion-free parameter-space visualization, we provided an extended version of hyper-slices which allows for investigating the different partitions and their boundaries. For providing an overview, the segmentation embedding can be used, as well as a 2D projection of the parameter-space samples. Additional coordinated views allow for investigating the temporal evolution of all simulation runs as well as individual simulation runs. We applied our approach to three real-world datasets from different domains and discussed it with corresponding domain experts. They rated the visualizations as helpful and highlighted that our approach covers many facets of the data analysis and enables a broader spectrum of analysis tasks.

Besides analyzing partitionings of the parameter space, we also presented methods to analyze the dependencies of the simulation outcome on the input parameters. As simulations often result in complex data where specific features should be preserved, we developed an interactive workflow to define characteristic measures highly specific to the analysis goal (see Chapter 5). The key component of the approach includes directly embedding a programming interface for defining the measure in the visual analysis application. The practical applicability was shown by describing how to define an order parameter for state transitions in active crystals [101].

Defining new measures to reduce the dimensionality of the simulation outcome is especially helpful for data types and analysis tasks where no established measures exist. For scalar fields, on the other hand, topological descriptors are widely used. In Chapter 6, we discussed how topological landscapes can be used to study the evolution over parameter variations. Therefore, we extended the approach by Herick et al. [154] for temporally coherent topological landscapes. The application to 2D and 3D ensemble datasets showed that coherent topological landscapes allow for determining parameter values at which topological changes in the simulation output occur. Additionally, stacked visualizations allow for an overview of the topological variations over the parameter range.

While the previously discussed approaches study the influence of the input parameters on the complete simulation output, we also investigated spatial variations in the parameter sensitivities (see Chapter 7). Here, we proposed a novel visualization of multiple, spatially resolved sensitivity values. It is embedded in a workflow using coordinated views to analyze different aspects of the spatial sensitivities. The practical applicability was shown by applying the approach to the analysis of medical simulations and provided insights into which spatial regions are more or less sensitive to the input parameters.

The second part of the thesis targets the uncertainty-aware analysis of correlations among different spatial regions in simulation ensembles. We present methods for a global correlation analysis that covers all pair-wise correlations between different regions and does not require the choice of a reference point.

We proposed similarity images (see Chapter 8) that provide a static overview visualization by encoding correlations as perceived differences between colors. Thus, high correlations correspond to similar colors, while anti-correlations are encoded as different colors. Similarity images computed for a climate ensemble encode climate anomalies like the North Atlantic Warming Hole and the characteristic regions for the El Niño/Southern Oscillation. To investigate correlations between regions in more detail, we presented an interactive visual analysis approach covering the uncertainty and the time lags between the different time series (see Chapter 9). Based on a hierarchical segmentation of the similarity image, multiple coordinated views enable an analysis on multiple levels of detail. In discussion with a domain expert, he rated our tool as helpful for finding climate phenomena. As climate simulations commonly include multiple fields, we extended the approach for multi-field data (see Chapter 10). We use an 2D embedding of the segments to show all correlations between all spatial segments of all fields. Thus, groups of segments of various fields can be easily selected for further analysis by selecting clusters in the embedding. Visualizations on different levels of detail allow for a comprehensive analysis, including the variation over the ensemble and the Fourier spectrum. Analyzing a large climate ensemble enabled us to reproduce several observations from the IPCC report and climate science in general.

Most of the approaches presented in this thesis have been evaluated by discussing them with domain experts. While their specific feedback is discussed in the corresponding chapters, some general observations will be discussed here. First, collecting feedback early and regularly in the design and development process and, thus, closely collaborating with the domain experts is very important to address the analysis tasks of interest. For example, for the multi-field correlation analysis presented in Chapter 10, analyzing the frequencies of the time series for the corresponding segments was important for the domain scientist, which we were unaware of. Additionally, all domain scientists mentioned the importance of including visualizations that they are familiar with and want to visualize individual simulation outputs directly. This allows them to connect their new findings with previous knowledge of the data. This connection to previous knowledge is also strengthened by accurately labeling the data where adaptations to the customs of the domain are necessary. By starting the analysis with replicating previous knowledge, domain experts are able to build trust in the approach and become familiar with the visualizations.

Many of the proposed approaches depend on similarity measures of different kinds: The parameter-space partitioning depends on the similarity measure between scalar fields, and creating coherent topological landscapes is based on the similarity between merge trees or, more specifically, their nodes. The analysis of spatial correlations is based on the choice of the correlation measure, which is also used as a similarity measure to create the similarity images and the embedding for the multi-field analysis. Although it is possible to easily exchange the similarity measure in all approaches without significantly changing all the other steps, it should be chosen with great care. While a detailed comparison of different similarity measures could guide the user, often, this choice strongly depends on the analysis task.

While the methods presented in this thesis cover different analysis tasks, they all work on simulation ensembles, albeit with slightly different facets. That is why the practical applicability could benefit from a stronger integration of the approaches. For example, the coherent topological landscapes could be combined with the system for analyzing the parameter-space partitioning by studying the topological variation over a parameter value of a selected segment. Including the possibility to interactively define characteristic measures before partitioning the parameter space would allow us to be very flexible in the aspects of the simulation ensemble that should be included. The analysis of global hierarchical correlations could also be combined with the spatial sensitivity analysis as both approaches analyze the variations over space. This allows for identifying regions of high sensitivity to input parameters and then analyzing how these regions correlate to others, potentially also in multiple other fields.

Some of the research prototypes developed within the scope of this thesis are available as open source[1]. However, only providing the source code does not lead to a common application in the different domains. Bringing research prototypes to practice is a challenging task that has been discussed increasingly over the recent years, as demonstrated by the the VisGap workshop[2] dedicated to this problem. One option to bring the approaches presented in this thesis closer to the domain experts would be to include them in the tools they commonly use for analyzing the data, which requires a significant amount of implementation effort.

In this work, we presented a set of approaches targeted at parameter space and correlation analysis in ensemble data. However, this field still contains several open research directions. While we mainly used

---

[1] `https://github.com/marinaevers/`

[2] `https://visgap.gitlab.io`

synthetic datasets for verification and evaluated our approaches algorithmically and by collecting (informal) feedback from domain experts, a more perception-based evaluation might yield additional insights. Performing user studies on the systems provided in this thesis is difficult as they are aimed at experts in their fields. Finding enough participants for user studies is, therefore, very challenging. Further abstracting the tasks would allow for performing user studies on individual aspects like selected visual encodings. We briefly discussed the need for a user study for the segmentation embedding in Chapter 3. However, other methods could also benefit from this kind of evaluation. For example, we compare the visualizations of the hyper-slicer to PCP and SPLOM and argue why the latter cannot solve all of our tasks. Comparing these visualizations on a broader set of tasks would allow deriving guidelines on when to use the hyper-slicer and in which cases one of the other visualizations is preferable. Similar considerations could be applied to the visual encoding for the sensitivity volumes in Chapter 7.

Many different applications for deep learning in scientific data visualization have been discussed lately [359]. Different steps of the workflows presented in this thesis could be improved by including deep learning methods. An obvious choice would be using surrogate models to obtain a better sampling of the parameter space without the need to run many computationally expensive simulations. Another option would be using neural networks to learn similarity measures, similar to SimilarityNet [161].

Finally, several more general open research questions are related to the work presented in this thesis. In many approaches, we used a dimensionality reduction algorithm on discrete samples of continuous, possibly uncertain data. However, the dimensionality reduction algorithms do neither consider the continuity of the data nor their uncertainty. While some approaches like Continuous Representation of the Projected Attribute Space [233] for continuous projections and uncertainty-aware MDS [135] provide first steps in this direction, the problem is not fully solved, especially when considering non-linear dimensionality reduction techniques.

Another broader field of research that is relatively poorly understood is the investigation of multiple ensembles together. In Part iii of this thesis, we worked with the Max-Planck-Institute Grand Ensemble but only focused on one of the scenarios while completely neglecting the others. Including the different scenarios in our approaches is a non-trivial task, as they add an additional dimension to the ensemble data and impose additional and different analysis tasks.

Part V

APPENDIX

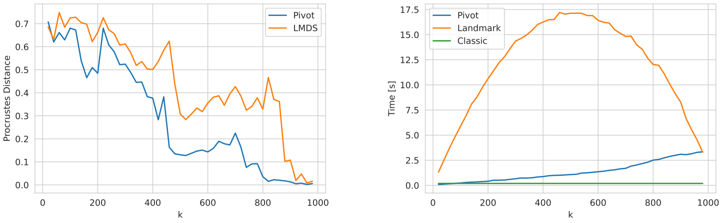
A

ADDITIONAL RESULTS

A.1 COMPARISON OF LMDS AND PMDS

For comparing the performance between LMDS and PMDS (see Section 2.3.1.2), we use a dataset of 100 data points with 10 dimensions with random coordinates. Note that both methods profit from similarity between the samples which is the reason why the random samples provide the worst case for the algorithms. For estimating the accuracy of the embedding, the Procrustes analysis [127] is used to determine the similarity to the embedding using classical MDS, which we consider as the ground truth. The Procrustes analysis can be used for determining the distance between two point clouds for the optimal scaling, rotation and translation. For the results shown here, we used the implementation provided by SciPy.

The results are shown in Figure A.1. It is relatively surprising that PMDS is faster than LMDS in most cases as theoretically, PMDS requires $\mathcal{O}(kn + k^2n)$ additional computations for n sample points and a subset of k samples. This observation might be explained by an inefficient com-



(a) Quality.

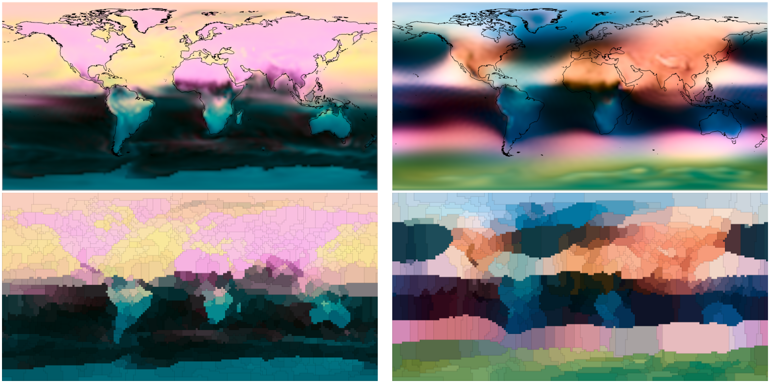
(b) Timings.

Figure A.1: Comparison between Landmark MDS and Pivot MDS on random synthetic data.

putation. Additionally, the dataset is relatively small such that classical MDS is in many cases faster than both algorithms. However, regarding the quality, we can observe that PMDS is indeed closer to the ground truth for most cases. Thus, we recommend the use of PMDS over LMDS.

A.2 SIMILARITY IMAGES

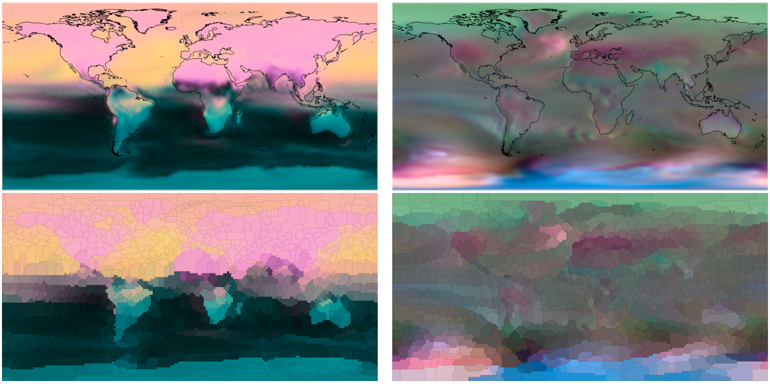
In the following, we present the similarity images and segmentations of the remaining fields of the MPI-GE ensemble (see Section 8.5). We also provide the percentages of the variation in the data that is covered in the 3D space (see Table A.1) and the reduction of the data that can be achieved by the segmentation (see Table A.2).



(a) ts.

(b) psl.

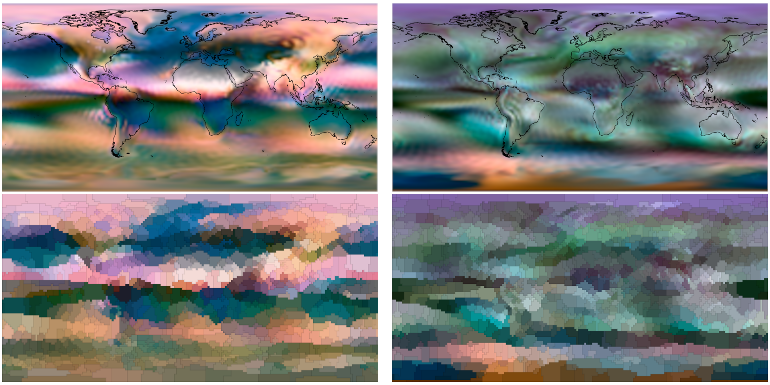
Figure A.2: Similarity images with coastlines and segmentations of the surface temperature and the sea level pressure.



(a) tas.

(b) tasAnomaly.

Figure A.3: Similarity images with coastlines and segmentations of the near-surface air temperature and its anomaly.



(a) pr.

(b) prAnomaly.

Figure A.4: Similarity images with coastlines and segmentations of the precipitation and its anomaly.

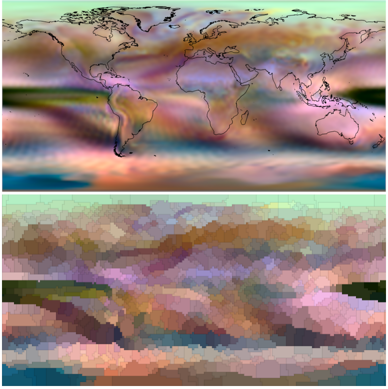


Figure A.5: Similarity image with coastlines and segmentation of the relative precipitation anomaly.

Table A.1: Variation of the dataset covered by the first three eigenvalues.

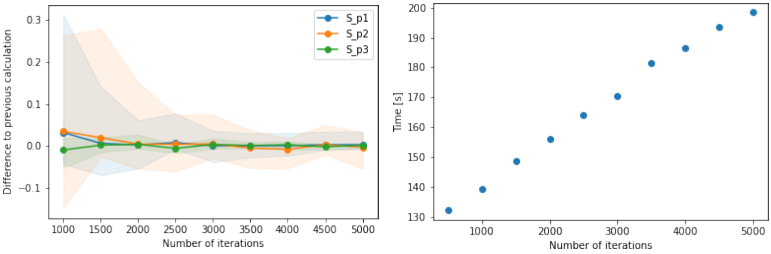
Field	**Variation**
Surface temperature (ts)	85.4%
Surface temperature anomaly (tsAnomaly)	26.7%
Near-surface air temperature (tas)	86.9%
Near-surface air temperature anomaly (tasAnomaly)	26.5%
Sea level pressure (psl)	53.4%
Sea level pressure anomaly (pslAnomaly)	36.6%
Precipitation (pr)	41.4%
Precipitation anomaly (prAnomaly)	9.0%
Relative precipitation anomaly (prRelativeAnomaly)	9.9%

Table A.2: Data reduction for the different fields from 18,432 initial samples.

Field	**Segments**	**% of original data**
tsAnomaly	1027	5.5718
prAnomaly	978	5.3059
psl	507	2.7507
ts	957	5.1921
prAnomaly	1075	5.8322
prRelativeAnomaly	980	5.3168
pslAnomaly	432	2.3438
tas	999	5.4199
tasAnomaly	902	4.8937

A.3 EVALUATION OF DGSA

In contrast to the other methods discussed in Section 7.5, DGSA also depends on input parameters which strongly influence the result. While we managed to evaluate the choice of the cluster number by an automatic screening procedure, it still depends on the number of iterations for the bootstrapping procedure. Therefore, we invesetigate the dependency on the number of iterations. Here, we observe a clear decrease in the variation but also an linear increase of the computation times as can be seen in Figure A.6. As a trade-off, we choose 4000 iterations.



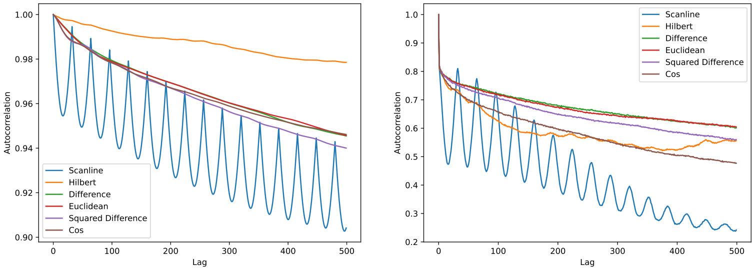
(a) Convergence (iterations).

(b) Timings (iterations).

Figure A.6: DGSA converges with an increase in the number of iterations (a) but the computation time also increases linearly (b).

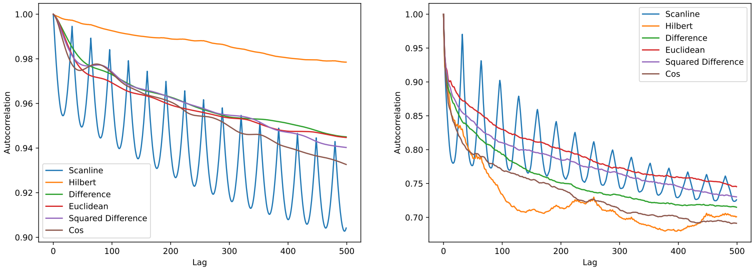
A.4 EVALUATION OF SPACE-FILLING CURVES

Figure A.7 presents the evaluation for the SFC (see Section 7.5.3) per dataset.



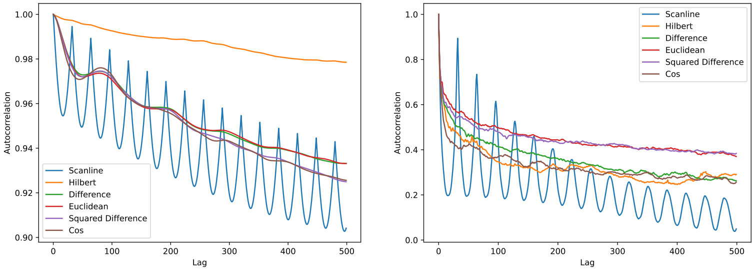
(a) Synthetic, positional coherence.

(b) Synthetic, value coherence.



(c) Ablation, positional coherence.

(d) Ablation, value coherence.



(e) Aneurysm, positional coherence.

(f) Aneurysm, value coherence.

Figure A.7: The autocorrelation of the sensitivity values reveals small differences between the datasets while the general trend agrees with the observations for the averaged data in Figure 7.15. The color map in a) applies to all visualizations except c).

B

DATASETS

B.1 SYNTHETIC DATASET WITH 4D PARAMETER SPACE

In the following, we will describe the creation of the synthetic dataset as used in Chapter 4 in more detail. The dataset is driven by four parameters a_1, a_2, a_3 and a_4 ranging from 0 to 1 leading to a 4D parameter space. The parameter space is sampled on a regular grid as well as using a Monte Carlo approach to allow for the comparison for other techniques. For both sampling methods, we choose 625 samples.

For each set of parameters, we create a 2D scalar field with Gaussians whose number depends on the parameters. The scalar field covers the domain $[0, 10] \times [0, 10]$ which is sampled with a resolution of 64×64. Independent of the parameter settings, one Gaussian is located in the center but its standard deviation depends on the parameter a_1. For $a_3 < 0.5$ a second Gaussian whose standard deviation also depends on a_3 is located in the top right corner. Further, the parameter space is split on the diagonal of the space formed by a_1, a_2 and a_3 where the third Gaussian in the lower left corner is only present on one half of the space. The parameter a_4 does not influence the result at all. In addition to this definition, we add noise in the range $[0, 0.1]$. Thus, the scalar field $g(\mathbf{x})$ is defined as

$$\begin{aligned} g(\mathbf{x}) =& \Theta(a_1 - a_2 - (1 - a_3)) \cdot f(\mathbf{x}; (1,7), 1) \\ &+ \Theta(0.5 - a_3) \cdot f(\mathbf{x}; (9,1), a_3 + 1) \\ &+ f(\mathbf{x}; (5,5), 0.1a_1 + 1) + \zeta, \end{aligned}$$

where $\Theta(z)$ is the Heavyside function with $\Theta(z) = 0$ for $z < 0$ and $\Theta(z) = 1$ otherwise, $f(\mathbf{x}; (x_1, x_2), \sigma)$ describes a 2D Gaussian kernel centered at (x_1, x_2) with standard deviation σ and ζ denotes uniform random noise in the range $[0, 0.1]$.

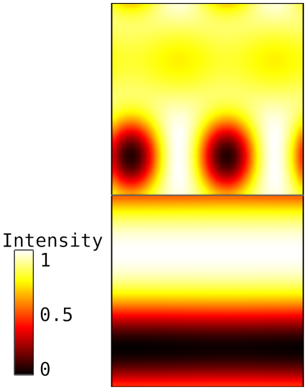


Figure B.1: The Swift-Hohenberg equation describes pattern formation in two dimensions (top: points, bottom: stripes).

B.2 REACTION-DIFFUSION SYSTEM

The reaction diffusion system as used in Chapter 6 is described by the partial differential equations [390]

$$\frac{\partial u_i}{\partial t} = d_{u_i} \Delta u_i + \alpha(u_j - u_i) + a - (b+1)u_i + u_i^2 v_i \tag{B.1}$$

and

$$\frac{\partial v_i}{\partial t} = d_{v_i} \Delta v_i + \alpha(v_j - v_i) + b u_i - u_i^2 v_i \,, \tag{B.2}$$

with $i, j \in [1, 2], i \neq j$ and where u_1, u_2, v_1, v_2 are the four components over a two-dimensional domain and varying over time. The parameters a, b, α, d_{u_1}, d_{u_2}, d_{v_1} and d_{v_2} influence which spatial pattern will form and their influence should be investigated. The input parameters influence the type of pattern strongly. Two example runs with stripe and point patterns are shown in Figure B.1. For the analysis in this work, we focus on the field u_1 with a spatial resolution of 32×32.

B.3 CAVITY FLOW

The lid-driven cavity flow is a popular benchmark problem in flow simulation [401]. In Chapter 6, we consider a 3D domain with unit length and use the Navier-Stokes equation for modelling the flow. The geometry is shown in Figure B.2, where the top part of the box is moved with a contant dimensionless velocity of 1 while on all other boundaries

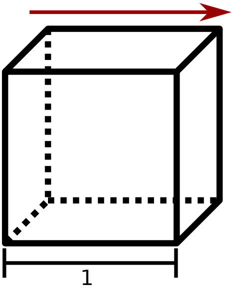


Figure B.2: The lid-driven cavity flow problem models the flow in a box where the top lid is moving with a constant speed.

no-slip boundary conditions are used. For the simulation, we use FEniCSx [307, 306, 10].

The flow is driven by the Reynolds number Re which is a dimensionless number that determines the transition between laminar and turbulent flow. We vary the Reynolds number between 100 and 1000 in steps of 100. Thus, we obtain a simulation ensemble consisting of 10 different ensemble members. Each simulation run consists of 99 steps with a dimensionless step width of 0.01. We omit the first time step where no flow is present yet. For the analysis in Section 6.5.3, we consider only the velocity magnitude with a resolution of $32 \times 32 \times 32$.

BIBLIOGRAPHY

[1] S. Afzal, R. Maciejewski, and D. S. Ebert. Visual analytics decision support environment for epidemic modeling and response evaluation. In *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 191–200, 2011.

[2] A. Agarwal, L. Caesar, N. Marwan, R. Maheswaran, B. Merz, and J. Kurths. Network-based identification and characterization of teleconnections on different scales. *Scientific Reports*, 9(1):1–12, 2019.

[3] M. Ahmed, Z. Liu, S. Humphries, and S. N. Goldberg. Computer modeling of the combined effects of perfusion, electrical conductivity, and thermal conductivity on tissue heating patterns in radiofrequency tumor ablation. *International Journal of Hyperthermia*, 24(7):577–588, 2008.

[4] J. Ahrens, B. Geveci, and C. Law. Paraview: An end-user tool for large data visualization. *The visualization handbook*, 717(8), 2005.

[5] W. Aigner, S. Miksch, W. Muller, H. Schumann, and C. Tominski. Visual methods for analyzing time-oriented data. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):47–60, 2008.

[6] 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.

[7] H. Akiba, N. Fout, and K.-L. Ma. Simultaneous classification of time-varying volume data based on the time histogram. In *EuroVis*, volume 6, pages 1–8. Citeseer, 2006.

[8] O. S. Alabi, X. Wu, J. M. Harter, M. Phadke, L. Pinto, H. Petersen, S. Bass, M. Keifer, S. Zhong, and C. Healey. Comparative visualization of ensembles using ensemble surface slicing. In *Visualization and Data Analysis 2012*, volume 8294, pages 318–329. SPIE, 2012.

[9] M. J. Alam, T. Biedl, S. Felsner, M. Kaufmann, S. G. Kobourov, and T. Ueckerdt. Computing cartograms with optimal complexity. In *Proceedings of the twenty-eighth annual Symposium on Computational Geometry*, pages 21–30, 2012.

[10] M. S. Alnaes, A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software*, 40, 2014.

[11] B. Alper, N. Riche, G. Ramos, and M. Czerwinski. Design study of linesets, a novel set visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2259–2267, 2011.

[12] A. Amirkhanov, I. Kosiuk, P. Szmolyan, A. Amirkhanov, G. Mistelbauer, M. E. Gröller, and R. G. Raidou. ManyLands: A journey across 4D phase space of trajectories. In *Computer Graphics Forum*, volume 38, pages 191–202, 2019.

[13] G. Andrienko, N. Andrienko, S. Bremm, T. Schreck, T. V. Landesberger, P. Bak, and D. Keim. Space-in-time and time-in-space self-organizing maps for exploring spatiotemporal patterns. *Computer Graphics Forum*, 29(3):913–922, 2010.

[14] N. Andrienko and G. Andrienko. *Exploratory analysis of spatial and temporal data: a systematic approach*. Springer Science & Business Media, 2006.

[15] N. Andrienko and G. Andrienko. Visual analytics of movement: An overview of methods, tools and procedures. *Information Visualization*, 12(1):3–24, 2013.

[16] F. J. Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1):17–21, 1973.

[17] A. Antonov, G. Lohmann, M. Ionita, M. Dima, and L. Linsen. An interactive visual analysis tool for investigating teleconnections in climate simulations. *Environmental Earth Sciences*, 78(10):294, 2019.

[18] D. Asimov. The grand tour: a tool for viewing multidimensional data. *SIAM journal on scientific and statistical computing*, 6(1):128–143, 1985.

[19] T. Athawale and A. Entezari. Uncertainty quantification in linear interpolation for isosurface extraction. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2723–2732, 2013.

[20] T. Athawale, E. Sakhaee, and A. Entezari. Isosurface visualization of data with nonparametric models for uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):777–786, 2015.

[21] T. M. Athawale, D. Maljovec, L. Yan, C. R. Johnson, V. Pascucci, and B. Wang. Uncertainty visualization of 2D Morse complex ensembles using statistical summary maps. *IEEE Transactions on Visualization and Computer Graphics*, 28(4):1955–1966, 2020.

[22] C. Bajaj, V. Pascucci, and D. Schikore. The contour spectrum. In *Proceedings. Visualization '97*, pages 167–173, 1997.

[23] R. Ballester-Ripoll, G. Halter, and R. Pajarola. High-dimensional scalar function visualization using principal parameterizations. *The Visual Computer*, pages 1–18, 2023.

[24] R. Ballester-Ripoll, E. G. Paredes, and R. Pajarola. Tensor algorithms for advanced sensitivity metrics. *SIAM/ASA Journal on Uncertainty Quantification*, 6(3):1172–1197, 2018.

[25] R. Ballester-Ripoll, E. G. Paredes, and R. Pajarola. Sobol tensor trains for global sensitivity analysis. *Reliability Engineering & System Safety*, 183:311–322, 2019.

[26] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine learning*, 56(1-3):89–113, 2004.

[27] C. Batini, E. Nardelli, and R. Tamassia. A layout algorithm for data flow diagrams. *IEEE Transactions on Software Engineering*, (4):538–546, 1986.

[28] C. Bechinger, R. Di Leonardo, H. Löwen, C. Reichhardt, G. Volpe, and G. Volpe. Active particles in complex and crowded environments. *Reviews of Modern Physics*, 88(4):045006, 2016.

[29] M. Behrisch, B. Bach, N. Henry Riche, T. Schreck, and J.-D. Fekete. Matrix reordering methods for table and network visualization. In *Computer Graphics Forum*, volume 35, pages 693–716. Wiley Online Library, 2016.

[30] K. Beketayev, G. H. Weber, D. Morozov, A. Abzhanov, and B. Hamann. Geometry-preserving topological landscapes. In *Proceedings of the Workshop at SIGGRAPH Asia*, pages 155–160, 2012.

[31] K. Beketayev, D. Yeliussizov, D. Morozov, G. H. Weber, and B. Hamann. *Measuring the distance between merge trees*. Springer, 2014.

[32] J. Benesty, J. Chen, Y. Huang, and I. Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[33] W. Berger, H. Piringer, P. Filzmoser, and E. Gröller. Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction. *Computer Graphics Forum*, 30(3):911–920, 2011.

[34] S. Bergner, M. Sedlmair, T. Möller, S. N. Abdolyousefi, and A. Saad. Paraglide: Interactive parameter space partitioning for computer simulations. *IEEE Transactions on Visualization and Computer Graphics*, 19(9):1499–1512, 2013.

[35] P. Bille. A survey on tree edit distance and related problems. *Theoretical computer science*, 337(1-3):217–239, 2005.

[36] T. Bin Masood, J. Budin, M. Falk, G. Favelier, C. Garth, C. Gueunet, P. Guillou, L. Hofmann, P. Hristov, A. Kamakshidasan, C. Kappe, P. Klacansky, P. Laurin, J. Levine, J. Lukasczyk, D. Sakurai, M. Soler, P. Steneteg, J. Tierny, W. Usher, J. Vidal, and M. Wozniak. An Overview of the Topology ToolKit. In *TopoInVis*, 2019.

[37] A. Biswas, G. Lin, X. Liu, and H.-W. Shen. Visualization of time-varying weather ensembles across multiple resolutions. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):841–850, 2016.

[38] A. Bock, A. Pembroke, M. L. Mays, L. Rastaetter, T. Ropinski, and A. Ynnerman. Visual verification of space weather ensemble simulations. In *2015 IEEE Scientific Visualization Conference (SciVis)*, pages 17–24. IEEE, 2015.

[39] N. Boers, B. Goswami, A. Rheinwalt, B. Bookhagen, B. Hoskins, and J. Kurths. Complex networks reveal global pattern of extreme-rainfall teleconnections. *Nature*, 566(7744):373–377, 2019.

[40] G.-P. Bonneau, H.-C. Hege, C. R. Johnson, M. M. Oliveira, K. Potter, P. Rheingans, and T. Schultz. Overview and state-of-the-art of uncertainty visualization. In *Scientific Visualization*, pages 3–27. Springer, 2014.

[41] M. Booshehrian, T. Möller, R. M. Peterman, and T. Munzner. Vismon: Facilitating analysis of trade-offs, uncertainty, and sensitivity in fisheries management decision making. In *Computer Graphics Forum*, volume 31, pages 1235–1244. Wiley Online Library, 2012.

[42] U. D. Bordoloi, D. L. Kao, and H.-W. Shen. Visualization techniques for spatial probability density function data. *Data Science Journal*, 3:153–162, 2004.

[43] E. Borgonovo. A new uncertainty importance measure. *Reliability Engineering & System Safety*, 92(6):771–784, 2007.

[44] A. Borrelli and J. Wellmann. Computer simulations then and now: an introduction and historical reassessment. *NTM Zeitschrift für Geschichte der Wissenschaften, Technik und Medizin*, 27(4):407–417, 2019.

[45] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[46] P. Bovenkamp. *4D-Phasenkontrast- und Magnetisierungs-Sättigungstransfer-MRT zur Charakterisierung des vaskulären Systems*. PhD thesis, Westfälische Wilhelms-Universität Münster, 2016.

[47] J. M. Boyer and W. J. Myrvold. Simplified O(n) planarity by edge addition. *Graph Algorithms Appl*, 5:241, 2006.

[48] U. Brandes and C. Pich. Eigensolver methods for progressive multidimensional scaling of large data. In *International Symposium on Graph Drawing*, pages 42–53. Springer, 2006.

[49] R. Brecheisen, A. Vilanova, B. Platel, and B. ter Haar Romeny. Parameter sensitivity visualization for DTI fiber tracking. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1441–1448, 2009.

[50] P. Bremer, G. Weber, V. Pascucci, M. Day, and J. Bell. Analyzing and tracking burning structures in lean premixed hydrogen flames. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):248–260, 2010.

[51] C. A. Brewer. Color Brewer. `http://www.ColorBrewer.org`, 2013. Accessed 17-March-2020.

[52] K. Brodlie, R. Allendes Osorio, and A. Lopes. A review of uncertainty in data visualization. *Expanding the frontiers of visual analytics and visualization*, pages 81–109, 2012.

[53] S. Bruckner and T. Möller. Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1468–1476, 2010.

[54] J. Buchmüller, D. Jäckle, E. Cakmak, U. Brandes, and D. A. Keim. Motionrugs: Visualizing collective trends in space and time. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):76–86, 2018.

[55] H. Carr and D. Duke. Joint contour nets. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1100–1113, 2013.

[56] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry*, 24(2):75–94, 2003.

[57] D. Castelvecchi and E. Gibney. CERN makes bold push to build €21-billion supercollider. *Nature*, 2020.

[58] P. Chalermsook and A. Schmid. Finding triangles for maximum planar subgraphs. In *International Workshop on Algorithms and Computation*, pages 373–384. Springer, 2017.

[59] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[60] C.-K. Chen, C. Wang, K.-L. Ma, and A. T. Wittenberg. Static correlation visualization for large time-varying volume data. In *2011 IEEE Pacific Visualization Symposium*, pages 27–34. IEEE, 2011.

[61] D. Chen, M. Rojas, B. Samset, K. Cobb, A. Diongue Niang, P. Edwards, S. Emori, S. Faria, E. Hawkins, P. Hope, P. Huybrechts, M. Meinshausen, S. Mustafa, G.-K. Plattner, and A.-M. Tréguier. *Framing, Context, and Methods*, page 147–286. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, 2021.

[62] M. Chimani, C. Gutwenger, M. Jünger, G. W. Klau, K. Klein, and P. Mutzel. The open graph drawing framework (ogdf). *Handbook of graph drawing and visualization*, 2011:543–569, 2013.

[63] M. Chimani, M. Ilsen, and T. Wiedera. Star-struck by fixed embeddings: Modern crossing number heuristics. In *International Symposium on Graph Drawing and Network Visualization*, pages 41–56. Springer, 2021.

[64] B. Chopard and M. Droz. *Cellular Automata Modeling of Physical Systems*. Collection Alea-Saclay: Monographs and Texts in Statistical Physics. Cambridge University Press, 1998.

[65] M. C. Chuah and S. F. Roth. On the semantics of interactive visualizations. In *Proceedings IEEE Symposium on Information Visualization'96*, pages 29–36. IEEE, 1996.

[66] D. Collaris and J. J. van Wijk. ExplainExplore: Visual exploration of machine learning explanations. In *2020 IEEE Pacific Visualization Symposium (PacificVis)*, pages 26–35. IEEE, 2020.

[67] C. Collins, G. Penn, and S. Carpendale. Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1009–1016, 2009.

[68] K. A. Cook and J. J. Thomas. Illuminating the path: The research and development agenda for visual analytics. Technical report, Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2005.

[69] J. Cousty and L. Najman. Incremental algorithm for hierarchical minimum spanning forests and saliency of watershed cuts. In *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 272–283. Springer, 2011.

[70] J. Cousty, L. Najman, and B. Perret. Constructive links between some morphological hierarchies on edge-weighted graphs. In *Mathematical Morphology and Its Applications to Signal and Image Processing: 11th International Symposium, ISMM 2013, Uppsala, Sweden, May 27-29, 2013. Proceedings 11*, pages 86–97. Springer, 2013.

[71] P. Crossno. Challenges in visual analysis of ensembles. *IEEE Computer Graphics and Applications*, 38(2):122–131, 2018.

[72] R. Cutura, C. Morariu, Z. Cheng, Y. Wang, D. Weiskopf, and M. Sedlmair. Hagrid—gridify scatterplots with hilbert and gosper curves. In *Proceedings of the 14th International Symposium on Visual Information Communication and Interaction*, pages 1–8, 2021.

[73] O. Daae Lampe, C. Correa, K. Ma, and H. Hauser. Curve-centric volume reformation for comparative visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1235–1242, 2009.

[74] R. Dafner, D. Cohen-Or, and Y. Matias. Context-based space filling curves. In *Computer Graphics Forum*, volume 19, pages 209–218. Wiley Online Library, 2000.

[75] C. Dalelane, K. Winderlich, and A. Walter. Evaluation of global teleconnections in CMIP6 climate projections using complex networks. *Earth System Dynamics*, 14(1):17–37, 2023.

[76] D. Demir, K. Beketayev, G. H. Weber, P.-T. Bremer, V. Pascucci, and B. Hamann. Topology exploration with hierarchical landscapes. In *Proceedings of the Workshop at SIGGRAPH Asia*, WASA '12, page 147–154, New York, NY, USA, 2012. Association for Computing Machinery.

[77] I. Demir, C. Dick, and R. Westermann. Multi-charts for comparative 3D ensemble visualization. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2694–2703, 2014.

[78] I. Demir, M. Jarema, and R. Westermann. Visualizing the central tendency of ensembles of shapes. In *SIGGRAPH ASIA 2016 Symposium on Visualization*, pages 1–8, 2016.

[79] I. Demir, J. Kehrer, and R. Westermann. Screen-space silhouettes for visualizing ensembles of 3D isosurfaces. In *2016 IEEE Pacific Visualization Symposium (PacificVis)*, pages 204–208. IEEE, 2016.

[80] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph drawing*, volume 357. Prentice Hall, Upper Saddle River, NJ, 1999.

[81] W. Didimo, G. Liotta, and F. Montecchiani. A survey on graph drawing beyond planarity. *ACM Computing Surveys (CSUR)*, 52(1):1–37, 2019.

[82] H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+ context visualization of complex simulation data. In *VisSym*, volume 3, pages 239–248. Citeseer, 2003.

[83] H. Doraiswamy and V. Natarajan. Computing reeb graphs as a union of contour trees. *IEEE Transactions on Visualization and Computer Graphics*, 19(2):249–262, 2012.

[84] D. Dorling. Area cartograms: their use and creation. In *Concepts and techniques in modern geography series*. Environmental Publications, University of East Anglia, 1996.

[85] D. Drees, S. Leistikow, X. Jiang, and L. Linsen. Voreen–an open-source framework for interactive visualization and processing of large volume data. *arXiv preprint arXiv:2207.12746*, 2022.

[86] C. A. Duncan and M. T. Goodrich. Planar orthogonal and polyline drawing algorithms. In *Handbook of Graph Drawing and Visualization*, 2013.

[87] R. Earnshaw and N. Wiseman. *An introductory guide to scientific visualization*. Springer Science & Business Media, 2012.

[88] D. Eddelbuettel and R. Francois. *RInside: C++ Classes to Embed R in C++ Applications*, 2019. R package version 0.2.15.

[89] H. Edelsbrunner and J. Harer. Persistent homology-a survey. *Contemporary mathematics*, 453:257–282, 2008.

[90] H. Edelsbrunner and J. Harer. *Computational geometry: An introduction*. American Mathematical Soc., 2010.

[91] H. Edelsbrunner, J. Harer, A. Mascarenhas, V. Pascucci, and J. Snoeyink. Time-varying reeb graphs for continuous space-time data. *Computational Geometry: Theory and Applications, vol. 41, no. 3, November 1, 2008, pp. 149-166*, 41(LLNL-JRNL-403147), 2008.

[92] A. Efrat, Y. Hu, S. G. Kobourov, and S. Pupyrev. MapSets: Visualizing embedded and clustered graphs. In *International Symposium on Graph Drawing*, pages 452–463. Springer, 2014.

[93] C. Eichner, H. Schumann, and C. Tominski. Making parameter dependencies of time-series segmentation visually understandable. In *Computer Graphics Forum*, volume 39, pages 607–622. Wiley Online Library, 2020.

[94] N. Ekhtiari, C. Ciemer, C. Kirsch, and R. V. Donner. Coupled network analysis revealing global monthly scale co-variability patterns between sea-surface temperatures and precipitation in dependence on the ENSO state. *The European Physical Journal Special Topics*, 230(14):3019–3032, 2021.

[95] N. Elmqvist, P. Dragicevic, and J.-D. Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1539–1148, 2008.

[96] V. A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158, 1969.

[97] G. B. Ermentrout and L. Edelstein-Keshet. Cellular automata approaches to biological modeling. *Journal of Theoretical Biology*, 160(1):97–133, 1993.

[98] M. Espadoto, R. M. Martins, A. Kerren, N. S. Hirata, and A. C. Telea. Toward a quantitative survey of dimension reduction techniques. *IEEE Transactions on Visualization and Computer Graphics*, 27(3):2153–2173, 2019.

[99] M. Evers, K. Huesmann, and L. Linsen. Uncertainty-aware visualization of regional time series correlation in spatio-temporal ensembles. In *Computer Graphics Forum*, volume 40, pages 519–530. Wiley Online Library, 2021.

[100] M. Evers and L. Linsen. Multi-dimensional parameter-space partitioning of spatio-temporal simulation ensembles. *Computers & Graphics*, 104:140–151, 2022.

[101] M. Evers and R. Wittkowski. A colloidal time crystal and its tempomechanical properties. *arXiv preprint arXiv:2112.04498*, 2021.

[102] G. Favelier, N. Faraj, B. Summa, and J. Tierny. Persistence atlas for critical point variability in ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1152–1162, 2018.

[103] D. Fenwick, C. Scheidt, and J. Caers. Quantifying asymmetric parameter interactions in sensitivity analysis: application to reservoir modeling. *Mathematical Geosciences*, 46(4):493–511, 2014.

[104] O. Fernandes, S. Frey, G. Reina, and T. Ertl. Visual representation of region transitions in multi-dimensional parameter spaces. *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*, 2019.

[105] F. Ferstl, M. Kanzler, M. Rautenhaus, and R. Westermann. Visual analysis of spatial variability and global correlations in ensembles of iso-contours. In *Computer Graphics Forum*, volume 35, pages 221–230. Wiley Online Library, 2016.

[106] F. Ferstl, M. Kanzler, M. Rautenhaus, and R. Westermann. Time-hierarchical clustering and visualization of weather forecast ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):831–840, 2017.

[107] S. Few. Time on the horizon. `http://www.perceptualedge.com/articles/visual_business_intelligence/time_on_the_horizon.pdf`, 2008. [Online; accessed 25-March-2022].

[108] M. T. Fischer, A. Frings, D. A. Keim, and D. Seebacher. Towards a survey on static and dynamic hypergraph visualizations. In *2021 IEEE Visualization Conference (VIS)*, pages 81–85. IEEE, 2021.

[109] W. D. Fisher. On grouping for maximum homogeneity. *Journal of the American Statistical Association*, 53(284):789–798, 1958.

[110] A. Fofonov and L. Linsen. Multivisa: Visual analysis of multi-run physical simulation data using interactive aggregated plots. In *VISIGRAPP (3: IVAPP)*, pages 62–73, 2018.

[111] A. Fofonov and L. Linsen. Projected field similarity for comparative visualization of multi-run multi-field time-varying spatial data. *Computer Graphics Forum*, 38(1):286–299, 2018.

[112] A. Fofonov, V. Molchanov, and L. Linsen. Visual analysis of multi-run spatio-temporal simulations using isocontour similarity for projected views. *IEEE Transactions on Visualization and Computer Graphics*, (8):2037–2050, 2016.

[113] M. Franke, H. Martin, S. Koch, and K. Kurzhals. Visual analysis of spatio-temporal phenomena with 1D projections. In *Eurovis 2021-23rd EG Conference on Visualization*, 2021.

[114] S. Frey, F. Sadlo, and T. Ertl. Visualization of temporal similarity in field data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2023–2032, 2012.

[115] T. M. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[116] J. Fruth, O. Roustant, and T. Muehlenstaedt. The fanovagraph package: Visualization of interaction structures and construction of block-additive kriging models. *HAL preprint 00795229*, 2013.

[117] R. Fuchs and H. Hauser. Visualization of multi-variate scientific data. *Computer Graphics Forum*, 28(6):1670–1690, 2009.

[118] T. Fujiwara, N. Sakamoto, J. Nonaka, K. Yamamoto, and K.-L. Ma. A visual analytics framework for reviewing multivariate time-series data with dimensionality reduction. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1601–1611, 2020.

[119] E. R. Gansner, Y. Hu, and S. Kobourov. GMap: Visualizing graphs and clusters as maps. In *2010 IEEE Pacific Visualization Symposium (PacificVis)*, pages 201–208. IEEE, 2010.

[120] X. Gao, B. Xiao, D. Tao, and X. Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13:113–129, 2010.

[121] M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic Discrete Methods*, 4(3):312–316, 1983.

[122] H. Gibson, J. Faith, and P. Vickers. A survey of two-dimensional graph layout techniques for information visualisation. *Information Visualization*, 12(3-4):324–357, 2013.

[123] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289–309, 2011.

[124] J. Görtler, T. Spinner, D. Streeb, D. Weiskopf, and O. Deussen. Uncertainty-aware principal component analysis. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):822–831, 2019.

[125] L. Gosink, J. Anderson, W. Bethel, and K. Joy. Variable interactions in query-driven visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1400–1407, 2007.

[126] L. Gosink, K. Bensema, T. Pulsipher, H. Obermaier, M. Henry, H. Childs, and K. I. Joy. Characterizing and visualizing predictive uncertainty in numerical ensembles through bayesian model averaging. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2703–2712, 2013.

[127] J. C. Gower. Generalized procrustes analysis. *Psychometrika*, 40:33–51, 1975.

[128] H. Griethe and H. Schumann. The visualization of uncertain data: Methods and problems. In *SimVis*, pages 143–156, 2006.

[129] D. Günther, J. Salmon, and J. Tierny. Mandatory critical points of 2D uncertain scalar fields. In *Proceedings of the 16th Eurographics Conference on Visualization*, pages 31–40. Eurographics Association, 2014.

[130] D. Guo, J. Chen, A. MacEachren, and K. Liao. A visualization system for space-time and multivariate patterns (VIS-STAMP). *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1461–1474, 2006.

[131] A. Gyulassy, N. Kotava, M. Kim, C. D. Hansen, H. Hagen, and V. Pascucci. Direct feature visualization using Morse-Smale complexes. *IEEE Transactions on Visualization and Computer Graphics*, 18(9):1549–1562, 2011.

[132] A. Gyulassy, V. Natarajan, V. Pascucci, and B. Hamann. Efficient computation of Morse-Smale complexes for three-dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1440–1447, 2007.

[133] H. Ha, G. B. Kim, J. Kweon, S. J. Lee, Y.-H. Kim, D. H. Lee, D. H. Yang, and N. Kim. Hemodynamic measurement using four-dimensional phase-contrast MRI: quantification of hemodynamic parameters and clinical applications. *Korean journal of radiology*, 17(4):445–462, 2016.

[134] M. Hadwiger, C. Berger, and H. Hauser. High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *IEEE Visualization, 2003. VIS 2003.*, pages 301–308. IEEE, 2003.

[135] D. Hägele, T. Krake, and D. Weiskopf. Uncertainty-aware multidimensional scaling. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[136] D. M. Hamby. A review of techniques for parameter sensitivity analysis of environmental models. *Environmental Monitoring and Assessment*, 32(2):135–154, 1994.

[137] J. Han, H. Zheng, Y. Xing, D. Z. Chen, and C. Wang. V2V: A deep learning approach to variable-to-variable selection and translation for multivariate time-varying data. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1290–1300, 2020.

[138] L. Hao, C. G. Healey, and S. A. Bass. Effective visualization of temporal ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):787–796, 2016.

[139] W. Harvey and Y. Wang. Topological landscape ensembles for visualization of scalar-valued functions. In *Computer Graphics Forum*, volume 29, pages 993–1002. Wiley Online Library, 2010.

[140] H. Hauser, F. Ledermann, and H. Doleisch. Angular brushing of extended parallel coordinates. In *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.*, pages 127–130. IEEE, 2002.

[141] H. Hauser, L. Mroz, G. I. Bischi, and M. E. Groller. Two-level volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):242–252, 2001.

[142] S. Hazarika, H. Li, K. Wang, H. Shen, and C. Chou. NNVA: Neural network assisted visual analysis of yeast cell polarization simulation. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):34–44, 2020.

[143] S. Hazarika, A. Biswas, E. Lawrence, and P. J. Wolfram. Probabilistic principal component analysis guided spatial partitioning of multivariate ocean biogeochemistry data. In S. Dutta, K. Feige, K. Rink, and D. Zeckzer, editors, *Workshop on Visualisation in Environmental Sciences (EnvirVis)*. The Eurographics Association, 2021.

[144] S. Hazarika, A. Biswas, and H.-W. Shen. Uncertainty visualization using copula-based analysis in mixed distribution models. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):934–943, 2017.

[145] S. Hazarika, S. Dutta, and H.-W. Shen. Visualizing the variations of ensemble of isosurfaces. In *2016 IEEE Pacific Visualization Symposium (PacificVis)*, pages 209–213. IEEE, 2016.

[146] J. He, H. Chen, Y. Chen, X. Tang, and Y. Zou. Variable-based spatiotemporal trajectory data visualization illustrated. *IEEE Access*, 7:143646–143672, 2019.

[147] W. He, J. Wang, H. Guo, H.-W. Shen, and T. Peterka. CECAV-DNN: Collective ensemble comparison and visualization using deep neural networks. *Visual Informatics*, 2020.

[148] W. He, J. Wang, H. Guo, K.-C. Wang, H.-W. Shen, M. Raj, Y. S. G. Nashed, and T. Peterka. InSituNet: Deep image synthesis for parameter space exploration of ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):23–33, 2019.

[149] X. He, Y. Tao, Q. Wang, and H. Lin. A co-analysis framework for exploring multivariate scientific data. *Visual Informatics*, 2(4):254–263, 2018.

[150] X. He, Y. Tao, Q. Wang, and H. Lin. Multivariate spatial data visualization: A survey. *Journal of Visualization*, 22(5):897–912, 2019.

[151] K. Heimes, M. Evers, T. Gerrits, S. Gyawali, D. Sinden, T. Preusser, and L. Linsen. Studying the effect of tissue properties on radiofrequency ablation by visual simulation ensemble analysis. In R. G. Raidou, B. Sommer, T. W. Kuhlen, M. Krone, T. Schultz, and H.-Y. Wu, editors, *Eurographics Workshop on Visual Computing for Biology and Medicine*. The Eurographics Association, 2022.

[152] C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. De Floriani, G. Scheuermann, H. Hagen, and C. Garth. A survey of topology-based methods in visualization. In *Computer Graphics Forum*, volume 35, pages 643–667. Wiley Online Library, 2016.

[153] J. Heinrich and D. Weiskopf. State of the art of parallel coordinates. *Eurographics (State of the Art Reports)*, pages 95–116, 2013.

[154] M. Herick, V. Molchanov, and L. Linsen. Temporally coherent topological landscapes for time-varying scalar fields. In *VISIGRAPP (3: IVAPP)*, pages 54–61, 2020.

[155] I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[156] J. Herman and W. Usher. SALib: An open-source python library for sensitivity analysis. *The Journal of Open Source Software*, 2(9), 2017.

[157] D. Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Dritter Band: Analysis · Grundlagen der Mathematik· Physik Verschiedenes: Nebst Einer Lebensgeschichte*, pages 1–2, 1935.

[158] T. Höllt, A. Magdy, P. Zhan, G. Chen, G. Gopalakrishnan, I. Hoteit, C. D. Hansen, and M. Hadwiger. Ovis: A framework for visual analysis of ocean forecast ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1114–1126, 2014.

[159] T. Horak, P. Berger, H. Schumann, R. Dachselt, and C. Tominski. Responsive matrix cells: A focus+context approach for exploring and editing multivariate graphs. *IEEE Transactions on Visualization and Computer Graphics*, 27:1644–1654, 2020.

[160] Y. Hu, E. R. Gansner, and S. Kobourov. Visualizing graphs and clusters as maps. *IEEE Computer Graphics and Applications*, 30(6):54–66, 2010.

[161] K. Huesmann and L. Linsen. SimilarityNet: A deep neural network for similarity analysis within spatio-temporal ensembles. *Computer Graphics Forum*, 41(3):379–389, 2022.

[162] J. W. Hurrell, Y. Kushnir, G. Ottersen, and M. Visbeck. An overview of the North Atlantic oscillation. *Geophysical Monograph-American Geophysical Union*, 134:1–36, 2003.

[163] A. Inselberg and B. Dimsdale. Parallel coordinates. *Human-Machine Interactive Systems*, pages 199–233, 2009.

[164] T. Iwanaga, W. Usher, and J. Herman. Toward SALib 2.0: Advancing the accessibility and interpretability of global sensitivity analyses. *Socio-Environmental Systems Modelling*, 4:18155, 2022.

[165] D. Jäckle, F. Fischer, T. Schreck, and D. A. Keim. Temporal MDS plots for analysis of multivariate data. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):141–150, 2015.

[166] H. Jänicke, M. Böttinger, U. Mikolajewicz, and G. Scheuermann. Visual exploration of climate variability changes using wavelet analysis. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1375–1382, 2009.

[167] H. Jänicke, M. Böttinger, and G. Scheuermann. Brushing of attribute clouds for the visualization of multivariate data. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1459–1466, 2008.

[168] J. Jankowai and I. Hotz. Feature level-sets: Generalizing isosurfaces to multi-variate data. *IEEE Transactions on Visualization and Computer Graphics*, 26(2):1308–1319, 2020.

[169] M. Jarema, I. Demir, J. Kehrer, and R. Westermann. Comparative visual analysis of vector field ensembles. In *2015 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE, 2015.

[170] R. Jayakumar, K. Thulasiraman, and M. Swamy. O(n/sup 2/) algorithms for graph planarization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(3):257–267, 1989.

[171] D. Jen, P. Parente, J. Robbins, C. Weigle, R. M. Taylor, A. Burette, and R. Weinberg. Imagesurfer: A tool for visualizing correlations between two volume scalar fields. In *IEEE Visualization 2004*, pages 529–536. IEEE, 2004.

[172] A. Jena, U. Engelke, T. Dwyer, V. Raiamanickam, and C. Paris. Uncertainty visualisation: An interactive visual survey. In *2020 IEEE Pacific Visualization Symposium (PacificVis)*, pages 201–205. IEEE, 2020.

[173] J. Johansson and C. Forsell. Evaluation of parallel coordinates: Overview, categorization and guidelines for future research. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):579–588, 2015.

[174] J. Johansson, P. Ljung, M. Jern, and M. Cooper. Revealing structure within clustered parallel coordinates displays. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pages 125–132. IEEE, 2005.

[175] C. Johnson. Top scientific visualization research problems. *IEEE Computer Graphics and Applications*, 24(4):13–17, 2004.

[176] C. R. Johnson and A. R. Sanderson. A next step: Visualizing errors and uncertainty. *IEEE Computer Graphics and Applications*, 23(5):6–10, 2003.

[177] I. T. Jolliffe and J. Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A*, 374(2065):20150202, 2016.

[178] H. Jänicke, A. Wiebel, G. Scheuermann, and W. Kollmann. Multifield visualization using local statistical complexity. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1384–1391, 2007.

[179] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15, 1989.

[180] D. Kao, A. Luo, J. L. Dungan, and A. Pang. Visualizing spatially varying distribution data. In *Proceedings Sixth International Conference on Information Visualisation*, pages 219–225. IEEE, 2002.

[181] C. Kappe, M. Böttinger, and H. Leitte. Analysis of decadal climate predictions with user-guided hierarchical ensemble clustering. *Computer Graphics Forum*, 38(3):505–515, 2019.

[182] J. Kehrer, P. Filzmoser, and H. Hauser. Brushing moments in interactive visual analysis. In *Computer Graphics Forum*, volume 29, pages 813–822. Wiley Online Library, 2010.

[183] J. Kehrer and H. Hauser. Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):495–513, 2013.

[184] P. Keil, T. Mauritsen, J. Jungclaus, C. Hedemann, D. Olonscheck, and R. Ghosh. Multiple drivers of the North Atlantic warming hole. *Nature Climate Change*, 10(7):667–671, 2020.

[185] D. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann. *Mastering the information age: solving problems with visual analytics*. Eurographics Association, 2010.

[186] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler. Visual analytics: Scope and challenges. In *Visual data mining*, pages 76–90. Springer, 2008.

[187] K. C. Kempfert, Y. Wang, C. Chen, and S. W. K. Wong. A comparison study on nonlinear dimension reduction methods with kernel variations: Visualization, optimization and classification. *Intelligent Data Analysis*, 24(2):267–290, 2020.

[188] A. Kerren and F. Schreiber. Why integrate InfoVis and SciVis?: An example from systems biology. *IEEE Computer Graphics and Applications*, 34(6):69–73, 2014.

[189] L. Kettner, J. Rossignac, and J. Snoeyink. The safari interface for visualizing time-dependent volume data using iso-surfaces and contour spectra. *Computational Geometry*, 25(1-2):97–116, 2003.

[190] R. M. Kirby, H. Marmanis, and D. H. Laidlaw. Visualizing multivalued data from 2D incompressible flows using concepts from painting. In *Proceedings Visualization'99 (Cat. No. 99CB37067)*, pages 333–540. IEEE, 1999.

[191] G. W. Klau, K. Klein, and P. Mutzel. An experimental comparison of orthogonal compaction algorithms. In *Graph Drawing: 8th International Symposium, GD 2000 Colonial Williamsburg, VA, USA, September 20–23, 2000 Proceedings 8*, pages 37–51. Springer, 2001.

[192] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, and Jupyter development team. Jupyter notebooks - a publishing format for reproducible computational workflows. In F. Loizides and B. Scmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90, Netherlands, 2016. IOS Press.

[193] S. G. Kobourov, S. Pupyrev, and P. Simonetto. Visualizing graphs as maps with contiguous regions. In *EuroVis (Short Papers)*, 2014.

[194] W. Köpp and T. Weinkauf. Temporal merge tree maps: A topology-based static visualization for temporal scalar data. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1157–1167, 2022.

[195] P. Köthur, C. Witt, M. Sips, N. Marwan, S. Schinkel, and D. Dransch. Visual analytics for correlation-based comparison of time series ensembles. In *Computer Graphics Forum*, volume 34, pages 411–420. Wiley Online Library, 2015.

[196] M. Kraus. Visualization of uncertain contour trees. In *IMAGAPP/IVAPP*, pages 132–139, 2010.

[197] A. Kumpf, M. Rautenhaus, M. Riemer, and R. Westermann. Visual analysis of the temporal evolution of ensemble forecast sensitivities. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):98–108, 2018.

[198] A. Kumpf, J. Stumpfegger, P. F. Hartl, and R. Westermann. Visual analysis of multi-parameter distributions across ensembles of 3D fields. *IEEE Transactions on Visualization and Computer Graphics*, 2021.

[199] A. Kumpf, J. Stumpfegger, and R. Westermann. Cluster-based analysis of multi-parameter distributions in cloud simulation ensembles. In *VMV*, pages 73–80, 2019.

[200] A. Kumpf, B. Tost, M. Baumgart, M. Riemer, R. Westermann, and M. Rautenhaus. Visualizing confidence in cluster-based ensemble weather forecast analyses. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):109–119, 2017.

[201] P. Köthur, M. Sips, H. Dobslaw, and D. Dransch. Visual analytics for comparison of ocean model output with reference data: Detecting and analyzing geophysical processes using clustering ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1893–1902, 2014.

[202] M. Latif and N. S. Keenlyside. El Niño/Southern Oscillation response to global warming. *Proceedings of the National Academy of Sciences*, 106(49):20578–20583, 2009.

[203] S. Leistikow, K. Huesmann, A. Fofonov, and L. Linsen. Aggregated ensemble views for deep water asteroid impact simulations. *IEEE Computer Graphics and Applications*, 40(1):72–81, 2019.

[204] S. Leistikow, A. Nahardani, V. Hoerr, and L. Linsen. Interactive visual similarity analysis of measured and simulated multi-field tubular flow ensembles. In *VCBM*, pages 139–150, 2020.

[205] A. Lhuillier, C. Hurter, and A. Telea. State of the art in edge and trail bundling techniques. In *Computer Graphics Forum*, volume 36, pages 619–645. Wiley Online Library, 2017.

[206] A. E. Lie, J. Kehrer, and H. Hauser. Critical design and realization aspects of glyph-based 3D data visualization. In *Proceedings of the 25th Spring Conference on Computer Graphics*, pages 19–26, 2009.

[207] T. Liebmann, G. H. Weber, and G. Scheuermann. Hierarchical correlation clustering in multiple 2D scalar fields. In *Computer Graphics Forum*, volume 37, pages 1–12. Wiley Online Library, 2018.

[208] M. Lighthill. On the squirming motion of nearly spherical deformable bodies through liquids at very small reynolds numbers. *Communications on pure and applied mathematics*, 5(2):109–118, 1952.

[209] A. Ligmann-Zielinska and P. Jankowski. Spatially-explicit integrated uncertainty and sensitivity analysis of criteria weights in multicriteria land suitability evaluation. *Environmental Modelling & Software*, 57:235–247, 2014.

[210] L. Liu, A. P. Boone, I. T. Ruginski, L. Padilla, M. Hegarty, S. H. Creem-Regehr, W. B. Thompson, C. Yuksel, and D. H. House. Uncertainty visualization by representative sampling from prediction ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 23(9):2165–2178, 2016.

[211] R. Liu, H. Guo, and X. Yuan. A bottom-up scheme for user-defined feature comparison in ensemble data. In *SIGGRAPH Asia 2015 Visualization in High Performance Computing*, SA '15, New York, NY, USA, 2015. Association for Computing Machinery.

[212] R. Liu, H. Guo, and X. Yuan. User-defined feature comparison for vector field ensembles. *Journal of Visualization*, 20(2):217–229, 2017.

[213] S. Liu, W. Cui, Y. Wu, and M. Liu. A survey on information visualization: Recent advances and challenges. *The Visual Computer*, 30(12):1373–1393, 2014.

[214] S. Liu, D. Maljovec, B. Wang, P.-T. Bremer, and V. Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1249–1268, 2015.

[215] A.-P. Lohfink, F. Gartzky, F. Wetzels, L. Vollmer, and C. Garth. Time-varying fuzzy contour trees. In *2021 IEEE Visualization Conference (VIS)*, pages 86–90. IEEE, 2021.

[216] A.-P. Lohfink, F. Wetzels, J. Lukasczyk, G. H. Weber, and C. Garth. Fuzzy contour trees: Alignment and joint layout of multiple contour trees. *Computer Graphics Forum*, 2020.

[217] T. V. Long and L. Linsen. MultiClusterTree: Interactive visual exploration of hierarchical clusters in multidimensional multivariate data. *Computer Graphics Forum*, 28(3):823–830, 2009.

[218] S. López-Pintado and J. Romo. On the concept of depth for functional data. *Journal of the American Statistical Association*, 104(486):718–734, 2009.

[219] A. Love, A. Pang, and D. Kao. Visualizing spatial multivalue data. *IEEE Computer Graphics and Applications*, 25(3):69–79, 2005.

[220] M. Luboschik, M. Röhlig, A. T. Bittig, N. Andrienko, H. Schumann, and C. Tominski. Feature-driven visual analytics of chaotic parameter-dependent movement. *Computer Graphics Forum*, 34(3):421–430, 2015.

[221] M. Luboschik, S. Rybacki, F. Haack, and H.-J. Schulz. Supporting the integrated visual analysis of input parameters and simulation trajectories. *Computers & Graphics*, 39:37–47, 2014.

[222] B. Ma and A. Entezari. An interactive framework for visualization of weather forecast ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1091–1101, 2019.

[223] N. Maher, S. Milinski, L. Suarez-Gutierrez, M. Botzet, M. Dobrynin, L. Kornblueh, J. Kröger, Y. Takano, R. Ghosh, and C. Hedemann. The Max Planck Institute Grand Ensemble: Enabling the exploration of climate system variability. *Journal of Advances in Modeling Earth Systems*, 11(7):2050–2069, 2019.

[224] K. Matkovic, D. Gracanin, M. Jelovic, and H. Hauser. Interactive visual steering-rapid visual prototyping of a common rail injection

system. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1699–1706, 2008.

[225] K. Matkovic, D. Gracanin, B. Klarin, and H. Hauser. Interactive visual analysis of complex scientific data as families of data surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1351–1358, 2009.

[226] K. Matkovic, D. Gracanin, R. Splechtna, M. Jelovic, B. Stehno, H. Hauser, and W. Purgathofer. Visual analytics for complex engineering systems: Hybrid visual steering of simulation ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1803–1812, 2014.

[227] F. McGee, M. Ghoniem, G. Melançon, B. Otjacques, and B. Pinaud. The state of the art in multilayer network visualization. In *Computer Graphics Forum*, volume 38, pages 125–149. Wiley Online Library, 2019.

[228] L. McInnes, J. Healy, N. Saul, and L. Großberger. UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018.

[229] W. Meulemans, N. H. Riche, B. Speckmann, B. Alper, and T. Dwyer. Kelpfusion: A hybrid set visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 19(11):1846–1858, 2013.

[230] M. Meyer, M. Sedlmair, P. S. Quinan, and T. Munzner. The nested blocks and guidelines model. *Information Visualization*, 14(3):234–249, 2015.

[231] J. Meyer-Spradow, T. Ropinski, J. Mensmann, and K. Hinrichs. Voreen: A rapid-prototyping environment for ray-casting-based volume visualizations. *IEEE Computer Graphics and Applications*, 29(6):6–13, 2009.

[232] M. Mihai and R. Westermann. Visualizing the stability of critical points in uncertain scalar fields. *Computers & Graphics*, 41:13–25, 2014.

[233] V. Molchanov, A. Fofonov, and L. Linsen. Continuous representation of projected attribute spaces of multifields over any spatial sampling. In *Computer Graphics Forum*, volume 32, pages 301–310. Wiley Online Library, 2013.

[234] M. D. Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2):161–174, 1991.

[235] T. Munzner. Process and pitfalls in writing information visualization research papers. In *Information Visualization*, pages 134–153. Springer, 2008.

[236] T. Munzner. A nested model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928, 2009.

[237] T. Munzner. *Visualization analysis and design*. CRC press, 2014.

[238] L. Najman, J. Cousty, and B. Perret. Playing with Kruskal: Algorithms for morphological trees in edge-weighted graphs. In *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 135–146. Springer, 2013.

[239] National Research Council. *Assessing the reliability of complex models: mathematical and statistical foundations of verification, validation, and uncertainty quantification*. National Academies Press, 2012.

[240] C. Neuhauser, M. Hieronymus, M. Kern, M. Rautenhaus, A. Oertel, and R. Westermann. Visual analysis of model parameter sensitivities along warm conveyor belt trajectories using met. 3d (1.6. 0-multivar0). *Geoscientific Model Development Discussions*, 2023:1–34, 2023.

[241] B. D. Q. Nguyen, R. Hewett, and T. Dang. Visual features for multivariate time series. In *Proceedings of the 11th International Conference on Advances in Information Technology*, pages 1–8, 2020.

[242] T. Nocke, S. Buschmann, J. F. Donges, N. Marwan, H.-J. Schulz, and C. Tominski. Visual analytics of climate networks. *Nonlinear Processes in Geophysics*, 22(5):545, 2015.

[243] T. Nocke, M. Flechsig, and U. Bohm. Visual exploration and evaluation of climate-related simulation data. In *2007 Winter Simulation Conference*, pages 703–711. IEEE, 2007.

[244] T. Nocke, H. Schumann, and U. Böhm. Methods for the visualization of clustered climate data. *Computational Statistics*, 19(1):75–94, 2004.

[245] M. Novotny and H. Hauser. Outlier-preserving focus+ context visualization in parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):893–900, 2006.

[246] S. Nusrat and S. Kobourov. The state of the art in cartograms. In *Computer Graphics Forum*, volume 35, pages 619–642. Wiley Online Library, 2016.

[247] H. Obermaier, K. Bensema, and K. I. Joy. Visual trends analysis in time-varying ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 22(10):2331–2342, 2015.

[248] H. Obermaier and K. I. Joy. Future challenges for ensemble visualization. *IEEE Computer Graphics and Applications*, 34(3):8–11, 2014.

[249] Observable. Explore, analyze, and explain data. As a team. `https://observablehq.com/`. [Online; accessed 9-March-2022].

[250] P. Oesterling, C. Heine, H. Jänicke, and G. Scheuermann. Visual analysis of high dimensional point clouds using topological landscapes. In *2010 IEEE Pacific Visualization Symposium (PacificVis)*, pages 113–120, 2010.

[251] P. Oesterling, C. Heine, H. Janicke, G. Scheuermann, and G. Heyer. Visualization of high-dimensional point clouds using their density distribution's topology. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1547–1559, 2011.

[252] P. Oesterling, C. Heine, G. H. Weber, D. Morozov, and G. Scheuermann. Computing and visualizing time-varying merge trees for high-dimensional data. In *Topological Methods in Data Analysis and Visualization*, pages 87–101. Springer, 2017.

[253] D. Orban, D. F. Keefe, A. Biswas, J. Ahrens, and D. Rogers. Drag and track: A direct manipulation interface for contextualizing data instances within a continuous parameter space. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):256–266, 2019.

[254] A. T. Pang, C. M. Wittenbrink, and S. K. Lodha. Approaches to uncertainty visualization. *The Visual Computer*, 13(8):370–390, 1997.

[255] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. Multi-resolution computation and presentation of contour trees. In *Proc. IASTED Conference on Visualization, Imaging, and Image Processing*, pages 452–290. Citeseer, 2004.

[256] G. Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, 36(1):157–160, 1890.

[257] B. Perret, G. Chierchia, J. Cousty, S. J. F. Guimaraes, Y. Kenmochi, and L. Najman. Higra: Hierarchical graph analysis. *SoftwareX*, 10:1–6, 2019.

[258] T. Pfaffelmoser, M. Mihai, and R. Westermann. Visualizing the variability of gradients in uncertain 2D scalar fields. *IEEE Transactions on Visualization and Computer Graphics*, 19(11):1948–1961, 2013.

[259] T. Pfaffelmoser, M. Reitinger, and R. Westermann. Visualizing the positional and geometrical variability of isosurfaces in uncertain scalar fields. In *Computer Graphics Forum*, volume 30, pages 951–960. Wiley Online Library, 2011.

[260] T. Pfaffelmoser and R. Westermann. Visualization of global correlation structures in uncertain 2D scalar fields. In *Computer Graphics Forum*, volume 31, pages 1025–1034. Wiley Online Library, 2012.

[261] T. Pfaffelmoser and R. Westermann. Correlation visualization for structural uncertainty analysis. *International Journal for Uncertainty Quantification*, 3(2), 2013.

[262] T. Pfaffelmoser and R. Westermann. Visualizing contour distributions in 2D ensemble data. In *EuroVis (Short Papers)*, 2013.

[263] M. N. Phadke, L. Pinto, O. Alabi, J. Harter, R. M. Taylor II, X. Wu, H. Petersen, S. A. Bass, and C. G. Healey. Exploring ensemble visualization. In *Visualization and Data Analysis 2012*, volume 8294, page 82940B. International Society for Optics and Photonics, 2012.

[264] F. Pianosi, K. Beven, J. Freer, J. W. Hall, J. Rougier, D. B. Stephenson, and T. Wagener. Sensitivity analysis of environmental models: A systematic review with practical workflow. *Environmental Modelling & Software*, 79:214–232, 2016.

[265] N. Piccolotto, M. Bögl, and S. Miksch. Visual parameter space exploration in time and space. *Computer Graphics Forum*, 2023.

[266] H. Piringer, W. Berger, and J. Krasser. HyperMoVal: Interactive visual validation of regression models for real-time simulation. *Computer Graphics Forum*, 29(3):983–992, 2010.

[267] H. Piringer, S. Pajer, W. Berger, and H. Teichmann. Comparative visual analysis of 2D function ensembles. *Computer Graphics Forum*, 31(3pt3):1195–1204, 2012.

[268] J. Platt. FastMap, MetricMap, and Landmark MDS are all Nystrom algorithms. In *AISTATS*, 2005.

[269] Plotly Technologies Inc. Collaborative data science, 2015. `https://plot.ly`.

[270] J. Poco, A. Dasgupta, Y. Wei, W. Hargrove, C. Schwalm, R. Cook, E. Bertini, and C. Silva. Similarityexplorer: A visual intercomparison tool for multifaceted climate data. In *Computer Graphics Forum*, volume 33, pages 341–350. Wiley Online Library, 2014.

[271] M. Pont, J. Vidal, J. Delon, and J. Tierny. Wasserstein distances, geodesics and barycenters of merge trees. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):291–301, 2021.

[272] M. Pont, J. Vidal, and J. Tierny. Principal geodesic analysis of merge trees (and persistence diagrams). *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[273] K. Pothkow and H.-C. Hege. Positional uncertainty of isocontours: Condition analysis and probabilistic measures. *IEEE Transactions on Visualization and Computer Graphics*, 17(10):1393–1406, 2010.

[274] K. Pöthkow and H.-C. Hege. Nonparametric models for uncertainty visualization. In *Computer Graphics Forum*, volume 32, pages 131–140. Wiley Online Library, 2013.

[275] K. Potter, J. Kniss, R. Riesenfeld, and C. R. Johnson. Visualizing summary statistics and uncertainty. In *Computer Graphics Forum*, volume 29, pages 823–832. Wiley Online Library, 2010.

[276] K. Potter, P. Rosen, and C. R. Johnson. From quantification to visualization: A taxonomy of uncertainty visualization approaches. In *IFIP Working Conference on Uncertainty Quantification*, pages 226–249. Springer, 2011.

[277] K. Potter, A. Wilson, P.-T. Bremer, D. Williams, C. Doutriaux, V. Pascucci, and C. R. Johnson. Ensemble-vis: A framework for the statistical visualization of ensemble data. In *2009 IEEE International Conference on Data Mining Workshops*, pages 233–240. IEEE, 2009.

[278] B. Preim and D. Bartz. *Visualization in medicine: theory, algorithms, and applications*. Elsevier, 2007.

[279] F. Raith, G. Scheuermann, and C. Gillmann. Uncertainty-aware Detection and Visualization of Ocean Eddies in Ensemble Flow Fields - A Case Study of the Red Sea. In S. Dutta, K. Feige, K. Rink, and D. Zeckzer, editors, *Workshop on Visualisation in Environmental Sciences (EnvirVis)*. The Eurographics Association, 2021.

[280] R. Rao and S. K. Card. The table lens: merging graphical and symbolic representations in an interactive focus+ context visualization for tabular information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 318–322, 1994.

[281] M. Rautenhaus, M. Böttinger, S. Siemen, R. Hoffman, R. M. Kirby, M. Mirzargar, N. Röber, and R. Westermann. Visualization in meteorology—a survey of techniques and tools for data analysis tasks. *IEEE Transactions on Visualization and Computer Graphics*, 24(12):3268–3296, 2018.

[282] G. Reeb. Sur les points singuliers d'une forme de pfaff completement integrable ou d'une fonction numerique [On the singular points of a completely integrable pfaff form or of a numerical function]. *Comptes Rendus Acad. Sciences Paris*, 222:847–849, 1946.

[283] H. Reijner. The development of the horizon graph. In *Proc. Vis08 Workshop From Theory to Practice: Design, Vision and Visualization*, volume 3, 2008.

[284] J. Ren, J. Schneider, M. Ovsjanikov, and P. Wonka. Joint graph layouts for visualizing collections of segmented meshes. *IEEE Transactions on Visualization and Computer Graphics*, 24(9):2546–2558, 2017.

[285] G. Ristovski, N. Garbers, H. K. Hahn, T. Preusser, and L. Linsen. Uncertainty-aware visual analysis of radiofrequency ablation simulations. *Computers & Graphics*, 79:24–35, 2019.

[286] G. Ristovski, T. Preusser, H. K. Hahn, and L. Linsen. Uncertainty in medical visualization: Towards a taxonomy. *Computers & Graphics*, 39:60–73, 2014.

[287] J. C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Fifth international conference on coordinated and multiple views in exploratory visualization (CMV 2007)*, pages 61–71. IEEE, 2007.

[288] R. C. Roberts, R. S. Laramee, G. A. Smith, P. Brookes, and T. D'Cruze. Smart brushing for parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 25(3):1575–1590, 2018.

[289] T. Ropinski and B. Preim. Taxonomy and usage guidelines for glyph-based medical visualization. In *SimVis*, volume 522, pages 121–138, 2008.

[290] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[291] D. Sacha, L. Zhang, M. Sedlmair, J. A. Lee, J. Peltonen, D. Weiskopf, S. C. North, and D. A. Keim. Visual interaction with dimensionality reduction: A structured literature analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):241–250, 2017.

[292] S. Sagiroglu and D. Sinanc. Big data: A review. In *2013 international conference on collaboration technologies and systems (CTS)*, pages 42–47. IEEE, 2013.

[293] H. Saikia, H.-P. Seidel, and T. Weinkauf. Extended branch decomposition graphs: Structural comparison of scalar data. In *Computer Graphics Forum*, volume 33, pages 41–50. Wiley Online Library, 2014.

[294] E. Sakhaee and A. Entezari. A statistical direct volume rendering framework for visualization of uncertain data. *IEEE Transactions on Visualization and Computer Graphics*, 23(12):2509–2520, 2016.

[295] S. Şalap-Ayça, P. Jankowski, K. C. Clarke, and A. Nara. Is less more? Experimenting with visual stacking of coincident maps for spatial global sensitivity analysis in urban land-use change modeling. *Environmental Modelling & Software*, 145:105181, 2021.

[296] P. Salembier and L. Garrido. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE Transactions on Image Processing*, 9(4):561–576, 2000.

[297] A. Saltelli. Making best use of model evaluations to compute sensitivity indices. *Computer physics communications*, 145(2):280–297, 2002.

[298] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[299] A. Saltelli, S. Tarantola, and K.-S. Chan. A quantitative model-independent method for global sensitivity analysis of model output. *Technometrics*, 41(1):39–56, 1999.

[300] J. Sanyal, S. Zhang, J. Dyer, A. Mercer, P. Amburn, and R. Moorhead. Noodles: A tool for visualization of numerical weather model ensemble uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1421–1430, 2010.

[301] N. Sauber, H. Theisel, and H.-P. Seidel. Multifield-graphs: An approach to visualizing correlations in multifield scalar data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):917–924, 2006.

[302] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[303] W. Schroeder, K. M. Martin, and W. E. Lorensen. *The visualization toolkit an object-oriented approach to 3D graphics*. Prentice-Hall, Inc., 1998.

[304] H.-J. Schulz and H. Schumann. Visualizing graphs-a generalized view. In *Tenth International Conference on Information Visualisation (IV'06)*, pages 166–173. IEEE, 2006.

[305] M. Schwarzl, L. Autin, G. Johnson, T. Torsney-Weir, and T. Möller. Cellpackexplorer: Interactive model building for volumetric data of complex cells. *Computers & Graphics: X*, 2:100010, 2019.

[306] M. W. Scroggs, I. A. Baratta, C. N. Richardson, and G. N. Wells. Basix: a runtime finite element basis evaluation library. *Journal of Open Source Software*, 7(73):3982, 2022.

[307] M. W. Scroggs, J. S. Dokken, C. N. Richardson, and G. N. Wells. Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes. *ACM Transactions on Mathematical Software*, 48(2):18:1–18:23, 2022.

[308] M. Sedlmair, C. Heinzl, S. Bruckner, H. Piringer, and T. Möller. Visual parameter space analysis: A conceptual framework. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2161–2170, 2014.

[309] M. Sedlmair, M. Meyer, and T. Munzner. Design study methodology: Reflections from the trenches and the stacks. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2431–2440, 2012.

[310] T. W. Sheu, C. Chou, S. Tsai, and P. Liang. Three-dimensional analysis for radio-frequency ablation of liver tumor with blood perfusion effect. *Computer methods in biomechanics and biomedical engineering*, 8(4):229–240, 2005.

[311] N. Shi, J. Xu, H. Li, H. Guo, J. Woodring, and H.-W. Shen. Vdl-surrogate: A view-dependent latent-based model for parameter space exploration of ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[312] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on graphics (TOG)*, 11(1):92–99, 1992.

[313] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of IEEE Symposium on Visual Languages*, pages 336–343. IEEE, 1996.

[314] Q. Shu, H. Guo, J. Liang, L. Che, J. Liu, and X. Yuan. Ensemble-graph: Interactive visual analysis of spatiotemporal behaviors in ensemble simulation data. In *2016 IEEE Pacific Visualization Symposium (PacificVis)*, pages 56–63. IEEE, 2016.

[315] S. Sidwall Thygesen, T. B. Masood, M. Linares, V. Natarajan, and I. Hotz. Level of detail exploration of electronic transition ensembles using hierarchical clustering. *Computer Graphics Forum*, 2022.

[316] V. Silva and J. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. *Advances in neural information processing systems*, 15, 2002.

[317] I. Sobel and G. Feldman. A 3x3 isotropic gradient operator for image processing. *A talk at the Stanford Artificial Project*, pages 271–272, 1968.

[318] I. M. Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and computers in simulation*, 55(1-3):271–280, 2001.

[319] B.-S. Sohn and C. Bajaj. Time-varying contour topology. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):14–25, 2005.

[320] B. Spence, L. Tweedie, H. Dawkes, and H. Su. Visualization for functional design. In *Proceedings of the 1995 IEEE Symposium on Information Visualization*, INFOVIS '95, pages 4–10, Washington, DC, USA, 1995. IEEE Computer Society.

[321] R. Splechtna, K. Matkovic, D. Gracanin, M. Jelovic, and H. Hauser. Interactive visual steering of hierarchical simulation ensembles. In *2015 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE, 2015.

[322] R. Sridharamurthy, T. B. Masood, A. Kamakshidasan, and V. Natarajan. Edit distance between merge trees. *IEEE Transactions on Visualization and Computer Graphics*, 26(3):1518–1531, 2018.

[323] L. Stopar, P. Skraba, M. Grobelnik, and D. Mladenic. Streamstory: Exploring multivariate time series on multiple scales. *IEEE Transactions on Visualization and Computer Graphics*, 25(4):1788–1802, 2018.

[324] J. Stumpfegger, K. Höhlein, G. Craig, and R. Westermann. GPU accelerated scalable parallel coordinates plots. *Computers & Graphics*, 109:111–120, 2022.

[325] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.

[326] J. Sukharev, C. Wang, K.-L. Ma, and A. T. Wittenberg. Correlation study of time-varying multivariate climate data sets. In *2009 IEEE Pacific Visualization Symposium*, pages 161–168. IEEE, 2009.

[327] J. Sun, C. Wu, Y. Ge, Y. Li, and H. Yu. Spatial-temporal scientific data clustering via deep convolutional neural network. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3424–3429. IEEE, 2019.

[328] Y. Sun and M. G. Genton. Functional boxplots. *Journal of Computational and Graphical Statistics*, 20(2):316–334, 2011.

[329] U. H. Syeda, P. Murali, L. Roe, B. Berkey, and M. A. Borkin. Design study" lite" methodology: Expediting design studies and enabling the synergy of visualization pedagogy and social good. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020.

[330] A. Szymczak. Subdomain aware contour trees and contour evolution in time-dependent scalar fields. In *International Conference on Shape Modeling and Applications 2005 (SMI'05)*, pages 136–144. IEEE, 2005.

[331] R. Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.

[332] R. Tamassia. *Handbook of graph drawing and visualization*. CRC press, 2013.

[333] R. Tamassia, G. Di Battista, and C. Batini. Automatic graph drawing and readability of diagrams. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):61–79, 1988.

[334] J. Tao, M. Imre, C. Wang, N. V. Chawla, H. Guo, G. Sever, and S. H. Kim. Exploring time-varying multivariate volume data using matrix of isosurface similarity maps. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1236–1245, 2018.

[335] C. J. Taylor and D. J. Kriegman. Minimization on the Lie group SO (3) and related manifolds. *Yale University, Tech. Rep. 9405*, 1994.

[336] A. C. Telea. *Data visualization: principles and practice*. CRC Press, 2014.

[337] J. B. Tenenbaum, V. d. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[338] J. J. Thomas and K. A. Cook. A visual analytics agenda. *IEEE Computer Graphics and Applications*, 26(1):10–13, 2006.

[339] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. The topology toolkit. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):832–842, 2018.

[340] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. The Topology ToolKit. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, 2017. `https://topology-tool-kit.github.io/`.

[341] R. D. Torn and G. J. Hakim. Ensemble-based sensitivity analysis. *Monthly Weather Review*, 136(2):663–677, 2008.

[342] T. Torsney-Weir, T. Möller, M. Sedlmair, and R. M. Kirby. Hypersliceplorer: Interactive visualization of shapes in multiple dimensions. *Computer Graphics Forum*, 37(3):229–240, 2018.

[343] T. Torsney-Weir, A. Saad, T. Möller, H. Hege, B. Weber, J. Verbavatz, and S. Bergner. Tuner: Principled parameter finding for image segmentation algorithms using visual response surface exploration. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1892–1901, 2011.

[344] M. Tory and T. Möller. Human factors in visualization research. *IEEE Transactions on Visualization and Computer Graphics*, 10(1):72–84, 2004.

[345] A. Tyagi, Z. Cao, T. Estro, E. Zadok, and K. Mueller. ICE: An interactive configuration explorer for high dimensional categorical parameter spaces. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 23–34. IEEE, 2019.

[346] A. Unger, S. Schulte, V. Klemann, and D. Dransch. A visual analysis concept for the validation of geoscientific simulation models. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2216–2225, 2012.

[347] A. Unger and H. Schumann. Visual support for the understanding of simulation processes. In *2009 IEEE Pacific Visualization Symposium*, pages 57–64. IEEE, 2009.

[348] L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[349] L. van der Maaten, E. O. Postma, and H. J. van den Herik. Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, 10(66-71):13, 2009.

[350] S. van der Walt and N. Smith. Matplotlib colormaps. `https://bids.github.io/colormap/`. [Online; accessed 9-March-2022].

[351] M. Van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 212–220, 1997.

[352] J. J. Van Wijk and H. Van de Wetering. Cushion treemaps: Visualization of hierarchical information. In *Proceedings 1999 IEEE Symposium on Information Visualization (InfoVis' 99)*, pages 73–78. IEEE, 1999.

[353] J. J. van Wijk and R. van Liere. Hyperslice. In *Proceedings Visualization'93*, pages 119–125. IEEE, 1993.

[354] C. Vehlow, F. Beck, and D. Weiskopf. Visualizing group structures in graphs: A survey. In *Computer Graphics Forum*, volume 36, pages 201–225. Wiley Online Library, 2017.

[355] D. Vietinghoff, C. Heine, M. Böttinger, N. Maher, J. Jungclaus, and G. Scheuermann. Visual analysis of spatio-temporal trends in time-dependent ensemble data sets on the example of the North Atlantic Oscillation. In *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*, pages 71–80. IEEE, 2021.

[356] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[357] T. Von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, and D. W. Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. In *Computer Graphics Forum*, volume 30, pages 1719–1749. Wiley Online Library, 2011.

[358] A. Völker. Optische Abfrage der Wellenpaketdynamik in niederdimensionalen Halbleiterstrukturen: Eine theoretische Studie. Master's thesis, University of Münster, 2020.

[359] C. Wang and J. Han. DL4SciVis: A state-of-the-art survey on deep learning for scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[360] J. Wang, S. Hazarika, C. Li, and H.-W. Shen. Visualization and visual analysis of ensemble data: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 25(9):2853–2872, 2019.

[361] J. Wang, X. Liu, H.-W. Shen, and G. Lin. Multi-resolution climate ensemble parameter analysis with nested parallel coordinates plots. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):81–90, 2017.

[362] L. Wang, X. Tang, J. Zhang, and D. Guan. Correlation analysis for exploring multivariate data sets. *IEEE Access*, 6:44235–44243, 2018.

[363] M. O. Ward. XmdvTool: Integrating multiple methods for visualizing multivariate data. In *Proceedings Visualization'94*, pages 326–333. IEEE, 1994.

[364] M. O. Ward and J. Yang. Interaction spaces in data and information visualization. In *VisSym*, pages 137–145, 2004.

[365] J. H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical association*, 58(301):236–244, 1963.

[366] G. Weber, P.-T. Bremer, M. Day, J. Bell, and V. Pascucci. Feature tracking using reeb graphs. In *Topological Methods in Data Analysis and Visualization*, pages 241–253. Springer, 2011.

[367] G. Weber, P.-T. Bremer, and V. Pascucci. Topological landscapes: A terrain metaphor for scientific data. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1416–1423, 2007.

[368] G. H. Weber, P.-T. Bremer, and V. Pascucci. Topological cacti: Visualizing contour-based statistics. In *Topological Methods in Data Analysis and Visualization II*, pages 63–76. Springer, 2012.

[369] D. Weiskopf. Uncertainty visualization: Concepts, methods, and applications in biological data visualization. *Frontiers in Bioinformatics*, page 10, 2022.

[370] J. Weissenböck, B. Fröhler, E. Gröller, J. Kastner, and C. Heinzl. Dynamic volume lines: Visual comparison of 3D volumes through space-filling curves. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1040–1049, 2018.

[371] F. Wetzels and C. Garth. A deformation-based edit distance for merge trees. In *2022 Topological Data Analysis and Visualization (TopoInVis)*, pages 29–38. IEEE, 2022.

[372] F. Wetzels, H. Leitte, and C. Garth. Branch decomposition-independent edit distances for merge trees. In *Computer Graphics Forum*, volume 41, pages 367–378. Wiley Online Library, 2022.

[373] F. Wickelmaier. An introduction to MDS. *Sound Quality Research Unit, Aalborg University, Denmark*, 46(5):1–26, 2003.

[374] A. T. Wilson and K. C. Potter. Toward visual analysis of ensemble data sets. In *Proceedings of the 2009 Workshop on Ultrascale Visualization - UltraVis '09*, pages 48–53. ACM, ACM Press, 2009.

[375] S. Wolfram. Cellular automata. *Los Alamos Science*, pages 09–01, 1983.

[376] P. C. Wong, H. Foote, D. L. Kao, R. Leung, and J. Thomas. Multivariate visualization with data fusion. *Information Visualization*, 1(3-4):182–193, 2002.

[377] G. Woodin, B. Winter, and L. Padilla. Conceptual metaphor and graphical convention influence the interpretation of line graphs. *IEEE Transactions on Visualization and Computer Graphics*, 28(2):1209–1221, 2021.

[378] J. Woodring and H.-W. Shen. Chronovolumes: a direct rendering technique for visualizing time-varying data. In *Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics*, pages 27–34, 2003.

[379] 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.

[380] 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, 2008.

[381] F. Wu, G. Chen, J. Huang, Y. Tao, and W. Chen. EasyXplorer: A flexible visual exploration approach for multivariate spatial data. In *Computer Graphics Forum*, volume 34, pages 163–172. Wiley Online Library, 2015.

[382] H.-Y. Wu, M. Nollenburg, and I. Viola. Multi-level area balancing of clustered graphs. *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[383] K. Wu and S. Zhang. A contour tree based visualization for exploring data with uncertainty. *International Journal for Uncertainty Quantification*, 3(3), 2013.

[384] J. Wulms, J. Buchmüller, W. Meulemans, K. Verbeek, and B. Speckmann. Stable visual summaries for trajectory collections. In *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*, pages 61–70. IEEE, 2021.

[385] J. Xia, Y. Zhang, J. Song, Y. Chen, Y. Wang, and S. Liu. Revisiting dimensionality reduction techniques for visual cluster analysis: An empirical study. *IEEE Transactions on Visualization and Computer Graphics*, 2021.

[386] C. Xie, M. Li, H. Wang, and J. Dong. A survey on visual analysis of ocean data. *Visual Informatics*, 2019.

[387] E. Xu and H. Zhang. Spatially-explicit sensitivity analysis for land suitability evaluation. *Applied Geography*, 45:1–9, 2013.

[388] L. Yan, Y. Wang, E. Munch, E. Gasparovic, and B. Wang. A structural average of labeled merge trees for uncertainty visualization. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):832–842, 2019.

[389] H. Yang, R. Ballester-Ripoll, and R. Pajarola. SenVis: Interactive tensor-based sensitivity visualization. In *Computer Graphics Forum*, volume 40, pages 275–286. Wiley Online Library, 2021.

[390] L. Yang, M. Dolnik, A. M. Zhabotinsky, and I. R. Epstein. Spatial resonances and superposition patterns in a reaction-diffusion model with interacting Turing modes. *Physical Review Letters*, 88(20):208303, 2002.

[391] K.-H. Yeap and M. Sarrafzadeh. Floor-planning by graph dualization: 2-concave rectilinear modules. *SIAM Journal on Computing*, 22(3):500–526, 1993.

[392] J. S. Yi, Y. ah Kang, and J. Stasko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, 2007.

[393] C. Yuan and H. Yang. Research on k-value selection method of k-means clustering algorithm. *J*, 2(2):226–235, 2019.

[394] D. A. Zaitsev. A generalized neighborhood for cellular automata. *Theoretical Computer Science*, 666:21–35, 2017.

[395] H. Zhang, Y. Hou, D. Qu, and Q. Liu. Correlation visualization of time-varying patterns for multi-variable data. *IEEE Access*, 4:4669–4677, 2016.

[396] M. Zhang, L. Chen, Q. Li, X. Yuan, and J. Yong. Uncertainty-oriented ensemble data visualization and exploration using variable spatial spreading. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1808–1818, 2020.

[397] Y. Zhang, Y. Wang, and S. Parthasarathy. Visualizing attributed graphs via terrain metaphor. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data*

Mining, KDD '17, page 1325–1334, New York, NY, USA, 2017. Association for Computing Machinery.

[398] J. Zhao, F. Chevalier, E. Pietriga, and R. Balakrishnan. Exploratory analysis of time-series with chronolenses. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2422–2431, 2011.

[399] L. Zhou, C. R. Johnson, and D. Weiskopf. Data-driven space-filling curves. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1591–1600, 2020.

[400] L. Zhou and D. Weiskopf. Indexed-points parallel coordinates visualization of multivariate correlations. *IEEE Transactions on Visualization and Computer Graphics*, 24(6):1997–2010, 2017.

[401] O. C. Zienkiewicz, R. L. Taylor, and P. Nithiarasu. *The finite element method for fluid dynamics*. Butterworth-Heinemann, 2013.

LEBENSLAUF

Marina Evers,
geboren am 08.07.1995 in Lingen (Ems)

Schulbildung

2002-2006	Joseph-Tiesmeyer-Schule Emsbüren (Grundschule)
2006-2014	Franziskusgymnasium Lingen (Abitur 23.04.2014)

Studium

2014-2017	Bachelor of Science Physik, WWU Münster (Abschluss 31.7.2017)
2014-2018	Bachelor of Science Informatik, WWU Münster (Abschluss 26.07.2018)
2016-2017	Universidade de Lisboa, Portugal (Erasmus)
2017-2019	Master of Science Physik, WWU Münster (Abschluss 13.06.2019)
2019-2023	Promotion Informatik, WWU Münster

Tätigkeiten

2017-2019	Studentische Hilfskraft, WWU Münster
2019-2023	Wissenschaftliche Mitarbeiterin, WWU Münster

Beginn der Dissertation

07/2019	Institut für Informatik, WWU Münster, betreut durch Lars Linsen

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*. `classicthesis` is available for both LaTeX and LyX:

`https://bitbucket.org/amiede/classicthesis/`