

Comparative Uncertainty Visualization for High-Level Analysis of Scalar- and Vector-Valued Ensembles

Vom Fachbereich Informatik der Technischen Universität Kaiserslautern zur Verleihung des akademischen Grades Doktor der Naturwissenschaften (Dr. rer. nat.) genehmigte Dissertation

von

Anna-Pia Lohfink



Datum der wissenschaftlichen Aussprache: 6.Mai 2022

Dekan: Prof. Dr. Jens Schmitt

Berichterstatter: Prof. Dr. Christoph Garth

Berichterstatter: Prof. Dr. Gunther H. Weber



Scientific Visualization Lab
University of Kaiserslautern

DE-386

Abstract

With this thesis, I contribute to the research field of uncertainty visualization, considering parameter dependencies in multi valued fields and the uncertainty of automated data analysis. Like uncertainty visualization in general, both of these fields are becoming more and more important due to increasing computational power, growing importance and availability of complex models and collected data, and progress in artificial intelligence. I contribute in the following application areas:

Uncertain Topology of Scalar Field Ensembles. The generalization of topology-based visualizations to multi valued data involves many challenges. An example is the comparative visualization of multiple contour trees, complicated by the random nature of prevalent contour tree layout algorithms. I present a novel approach for the comparative visualization of contour trees –the Fuzzy Contour Tree.

Uncertain Topological Features in Time-Dependent Scalar Fields. Tracking features in time-dependent scalar fields is an active field of research, where most approaches rely on the comparison of consecutive time steps. I created a more holistic visualization for time-varying scalar field topology by adapting Fuzzy Contour Trees to the time-dependent setting.

Uncertain Trajectories in Vector Field Ensembles. Visitation maps are an intuitive and well-known visualization of uncertain trajectories in vector field ensembles. For large ensembles, visitation maps are not applicable, or only with extensive time requirements. I developed Visitation Graphs, a new representation and data reduction method for vector field ensembles that can be calculated in situ and is an optimal basis for the efficient generation of visitation maps. This is accomplished by bringing forward calculation times to the pre-processing.

Visually Supported Anomaly Detection in Cyber Security. Numerous cyber attacks and the increasing complexity of networks and their protection necessitate the application of automated data analysis in cyber security. Due to uncertainty in automated anomaly detection, the results need to be communicated to analysts to ensure appropriate reactions. I introduce a visualization system combining device readings and anomaly detection results: the Security in Process System. To further support analysts I developed an application agnostic framework that supports the integration of knowledge assistance and applied it to the Security in Process System. I present this Knowledge Rocks Framework, its application and the results of evaluations for both, the original and the knowledge assisted Security in Process System. For all presented systems, I provide implementation details, illustrations and applications.

Kurzfassung

Mit dieser Dissertation leiste ich einen Beitrag im Forschungsbereich der Unsicherheitsvisualisierung, insbesondere bezüglich Parameterabhängigkeiten in Ensembles und der Unsicherheit von automatischer Datenanalyse. Diese beiden Bereiche sind – wie Unsicherheitsvisualisierung im Allgemeinen – von zunehmender Bedeutung auf Grund steigender Rechenleistung, höherer Verfügbarkeit und Bedeutung komplexer Modelle und gespeicherter Daten und Fortschritten in der künstlichen Intelligenz. Ich trage in den folgenden Anwendungsbereichen bei:

Unsichere Topologie von Skalarfeld-Ensembles. Die Verallgemeinerung topologiebasierter Visualisierungen auf Ensembles stellt viele Herausforderungen. Ein Beispiel ist die vergleichende Visualisierung mehrerer Konturbäume die durch die zufällige Natur vorherrschender Layout Algorithmen verkompliziert wird. Ich beschreibe einen neuen Ansatz in diesem Bereich: Fuzzy Konturbäume.

Unsichere topologische Merkmale in zeitabhängigen Skalarfeldern. Das Verfolgen von Merkmalen zeitabhängiger Skalarfelder ist ein aktiver Forschungsbereich, in dem die meisten Ansätze auf den Vergleich aufeinanderfolgender Zeitschritte setzen. Durch die Anpassung von Fuzzy Konturbäumen an diese Felder habe ich eine umfassendere Visualisierung von zeitabhängiger Skalarfeldtopologie entwickelt.

Unsichere Trajektorien in Vektorfeld-Ensembles. Visitation Maps sind eine intuitive und verbreitete Visualisierung für unsichere Trajektorien in Vektorfeld-Ensembles. Im Falle großer Ensembles sind sie allerdings nicht oder nur mit großem Zeitaufwand anwendbar. Ich habe Visitation Graphs entwickelt, eine neue Darstellung und Datenreduzierungsmethode für Vektorfeld-Ensembles, die in situ berechnet werden kann und optimale Voraussetzungen für eine effiziente Visitation Map Erstellung bietet, indem Berechnungszeiten in die Vorverarbeitung verschoben werden.

Visuell Unterstützte Anomalieerkennung in der Cybersicherheit. Häufige Angriffe und die steigende Komplexität von Netzwerken und ihrer Absicherung machen die Anwendung automatisierter Datenanalyse notwendig. Die Ergebnisse der automatisierten Anomalieerkennung unterliegen Unsicherheiten und müssen deshalb an Analysten kommuniziert werden, um angemessene Reaktionen zu garantieren. Ich stelle ein Visualisierungssystem vor, das Messwerte und Ergebnisse der Anomalieerkennung kombiniert: das Security in Process System. Um Analysten weitergehend zu unterstützen, habe ich ein anwendungsunabhängiges Framework entwickelt, das die Integration von gespeichertem Wissen unterstützt und es auf das Security in Process System angewandt. Ich stelle dieses Knowledge Rocks Framework und seine Anwendung vor, sowie Ergebnisse von Nutzerstudien für das ursprüngliche und das wissensunterstützte Security in Process System. Für alle vorgestellten Systeme gebe ich Details bezüglich der Implementierung, Beispiele und Anwendungen.

Acknowledgement

I am deeply grateful to Christoph Garth for being always available and a reliable source of academic and general advice. You are the only person I know that (literally) runs across me and instantly turns a lack of motivation into excitement and a huge bunch of work.

Further, I would like to thank my husband Noël for support in long nights before paper deadlines where you stayed by my side. Everybody knows that without your help, my tools wouldn't have looked half as good.

For all further acknowledgments, I figured that a more abstract view would be a good choice for clarity and conciseness.

	SCIENTIFIC INPUT	DISTRACTION	EMOTIONAL SUPPORT	LOVE
HEIKE LEITTE	X			
GUNTHER WEBER	X			
MY KIDS		∇!∇!	X	X
MY FAMILY			X	X
CHRISTOPHER KAPPE	X	X		
ALL COAUTHORS	X			
JINX MONSOON'S MANTRA			X	

Thank you for influencing my work and my life.

This research was funded by the Deutsche Forschungsgemeinschaft (DFG – German Research Foundation) under contract 252408385 as part of IRTG 2057 Physical Modeling for Virtual Manufacturing.

Contents

1	Introduction	1
2	Background: Uncertainty and its Visualization	6
I	Parameter Dependency	11
3	Uncertain Topology of Scalar Field Ensembles	13
3.1	Review of Topology-Based Ensemble Visualization	15
3.1.1	Contour Trees	15
3.1.2	Visualization of Contour Trees	16
3.1.3	Topology-Based Ensemble Visualization	18
3.2	Tree Alignment of Contour Trees	20
3.2.1	Minimal Contour Tree Alignment	22
3.2.2	Alignment Heuristics	23
3.2.3	Cost Metrics	24
3.2.4	Algorithm	25
3.2.5	Properties of the Contour Tree Alignment	26
3.3	Fuzzy Contour Trees	27
3.3.1	Branch Decomposition of the Alignment	27
3.3.2	Layout Algorithm	29
3.3.3	Interaction	33
3.3.4	Layout Parameters	34
3.3.5	Challenges	35
3.4	Results	37
3.4.1	Analytical Ensemble with Outlier	38
3.4.2	Scattered Peaks Ensemble	41
3.4.3	Convection Simulation	43
3.4.4	Viscous Fingering	47
3.5	Discussion	48
3.6	Conclusion	49
4	Uncertain Topological Features in Time-Dependent Scalar Fields	51
4.1	Background: Topology Visualization on Time-Dependent Data	53
4.2	Tree Alignment of Time-Varying Contour Trees	54

4.3	Layout of Time-Varying Fuzzy Contour Trees	56
4.4	Interaction	59
4.4.1	The Time Selector	59
4.4.2	Interaction with Time-Varying Fuzzy Contour Tree	62
4.5	Results	63
4.5.1	Sea Ice	63
4.5.2	Convection Simulation	66
4.5.3	Cloud Top Pressure	66
4.6	Conclusion	68
5	Uncertain Trajectories in Vector Field Ensembles	69
5.1	Visualization of Uncertain Vector Fields: Background and Related Work	70
5.2	Visitation Graphs: Interactive Ensemble Visualization with Visitation Maps	76
5.2.1	Efficient Computation of Visitation Graphs	78
5.2.2	Efficient Approximation of Visitation Maps from Visitation Graphs	79
5.2.3	Space Requirements and Data Reduction	84
5.2.4	Application to Visualization	85
5.3	Experiments	88
5.4	Results	91
5.4.1	Industrial Stirring Simulation	92
5.4.2	Convection Simulation	93
5.4.3	Cavity Flow	94
5.5	Discussion and Conclusion	95
II	Uncertainty in Automated Data Analysis	99
6	Visually Supported Anomaly Detection in Cyber Security	101
6.1	Application Background and Challenges	104
6.1.1	Automated Anomaly Detection	105
6.1.2	State-of-the-art analysis	107
6.2	System Requirements	107
6.3	The Security in Process System	108
6.3.1	User Interface	109
6.3.2	Overview and Detail	110
6.3.3	Anomaly Highlighting	111
6.3.4	Interaction	112
6.3.5	Implementation and Scalability	113
6.4	Analysis Strategies and Usage Scenario	113
6.5	Expert Evaluation	119
6.6	Discussion and Conclusion	120

7	Knowledge Assistance for Decision-Making	122
7.1	Review of Knowledge Assistance in Visualization	123
7.1.1	The KAVA Model	124
7.1.2	Knowledge-Assisted Visualization Systems	126
7.2	The Knowledge Rocks Framework	127
7.2.1	Requirements	128
7.2.2	Architecture Definition	130
7.3	Implementation Steps for KAVA Processes with the Knowledge Rocks Framework	134
7.3.1	Application of the Framework	136
7.4	Discussion and Limitations.	139
7.5	Conclusion	140
8	The Knowledge Assisted Security in Process System	142
8.1	Knowledge Base	143
8.1.1	Acting Ontology	145
8.1.2	Suggesting Related Instances	151
8.2	Storing Instances	151
8.3	Visualization	153
8.4	Guidance	155
8.5	Data Size and Storage Access	155
8.6	Usage Scenarios and Expert Evaluation	156
8.7	Conclusion and Future Work	158
9	Evaluation of the Security in Process System	159
9.1	Experimental Design and Procedure	159
9.2	Tasks	160
9.3	Evaluation Setup	163
9.4	Results	165
9.4.1	Effectiveness	165
9.4.2	Satisfaction	167
9.4.3	Efficiency and Cognitive Load	169
9.5	Discussion of the Evaluation Results	170
III	Conclusion	173
10	Conclusion	174

” *The universe, they said, depended for its operation on the balance of four forces which they identified as charm, persuasion, uncertainty and bloody-mindedness.*

— **Sir Terence David John Pratchett**
(1986)

Not only on the discworld, but also in our world, uncertainty is omnipresent, especially when analyzing data. Uncertainty arises for example from multiple executions of experiments with differing results or inaccuracy of measurements, models and parameters. While uncertainty can not be prevented, it potentially has a huge impact. From hurricane warnings to surgery planning, uncertainty in the data can be a matter of life and death. Hence, the incorporation of uncertainty is an active field of research in visualization with many challenges [27].

There are Many Different Kinds of Uncertainty. Depending on the source of the uncertainty, whether it can be diminished or not, and how it can be quantified, there are different types of uncertainty that might need different treatment in the visualization – if they can be visualized at all. Distinguishing these uncertainty types and handling uncertainty in the data is a confusing obstacle in uncertainty visualization. Especially since each kind of uncertainty can be represented in different ways (as probability distribution function, collection of different results, ...).

Uncertainty Propagates. Figuring out what kind of uncertainty is present in the data and how to handle it, is –depending on the source of uncertainty and how exact the visualization should be– only the beginning of an uncertainty propagation chain. Every pass on the data can change and add uncertainty (Figure 1.1).

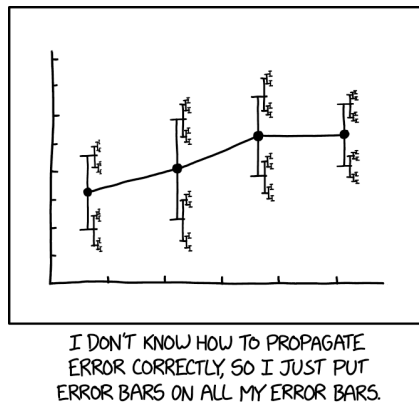


Fig. 1.1.: xkcd: error bars [140].

Uncertainty Visualization Requires Space. As can be seen in Figure 1.1, visualizing uncertainty variables adds (at least one) dimension to the visualization. Many visualization approaches without incorporated uncertainty already exploit all available visual properties. Adding uncertainty hence often requires substantial changes in the visualization, like the generation of additional geometry or changing the encoding of variables. Also, new communication channels like sound and animation might be necessary to integrate the additional information. Furthermore, with an increase of shown information, the risk of obfuscation and visual clutter increases, leading to the fact that many uncertainty visualization approaches tend to draw the attention to uncertain areas of the data, while in many approaches the more certain data should be emphasized [82].

Experts are needed. Finally, for a rigorous representation of uncertainty, expertise from statistics and potentially social sciences is required, which further complicates the development of such systems.

In my thesis, I contribute to the research on uncertainty visualization in different applications around comparative ensemble visualization. The visualized uncertainty is present in multi valued data or given as scalar value. To provide accessible support for analysts, I focus on high-level analysis, providing an overview of the data and its uncertainty. I support intuitive exploration with clear visual metaphors, leaving detailed uncertainty quantification and propagation to more specialized analysis approaches.

In addition to their individual applications and strengths, all visualization approaches I developed support users in **comparing**, **combining** and **separating** ensemble members and data intervals. The concrete importance of these basic visualization tasks in the respective applications is highlighted in the following chapters.

I describe visualization approaches considering parameter dependency in ensembles of model realizations (Part I), and a visualization system that communicates the uncertainty of automated data analysis with its enhancements (Part II).



Fuzzy Contour Trees. Fuzzy Contour Trees allow the combined visualization and comparison of the topology of different ensemble members. To do so, a mapping between the different topological structures and a thought-out layout algorithm that considers all members at once are required. This visualization contributes to the ongoing effort to make topology-based visualization approaches available for ensembles. I present Fuzzy Contour Trees in Chapter 3. Their adaption to time-varying scalar fields is introduced in Chapter 4. There, specific features for the time-varying setting support typical analysis tasks in time series analysis. As an adaption of an ensemble method, time-varying Fuzzy Contour Trees provide a higher flexibility in analysis and a more holistic view on the data than established approaches like for example feature tracking.



Visitation Graphs. I introduce Visitation Graphs to enable the application of visitation maps to ensembles and random fields that prohibit interactive runtimes using their naïve computation. Visitation Graphs are a representation of a vector field ensemble that can be calculated in situ and that provides an optimal basis for the interactive exploration of the vector field using visitation maps. Applying Visitation Graphs provides a tradeoff between generation time, calculation time for visitation maps, and accuracy. Using Visitation Graphs as a compressed representation of the vector field ensemble, visitation maps can be generated for ensembles that can not be entirely stored and hence completely prohibit the application of visitation maps. I present Visitation Graphs in Chapter 5.



The Security in Process System. Part II contains my work concerning uncertainty of automated data analysis. Automated data analysis is inevitable when facing insufficient resources in terms of time and/or analysts. Especially in cyber security, the required reaction times often make automated analysis inevitable. Yet, results of automated data analysis are uncertain and need to be revised by analysts before triggering a reaction based on the analysis results. To support this revision, we developed the Security in Process System. Specialized to the application in operational technology networks, it communicates sensor and actuator readings and the result of a new anomaly detection algorithm

combined in spiral plots. I present the Security in Process System in Chapter 6. Since triage analysis is a challenging task, especially when it is performed under time pressure, further support for cyber security analysts using the Security in Process System can improve their performance. To offer further support, we include knowledge assistance in the system. Adding knowledge assistance to an existing visualization system is a complex task and there are no generally applicable guidelines or frameworks that support it. To fill this gap, we developed the Knowledge Rocks framework. It provides a structure that, once implemented in a concrete application, provides all features that are necessary for knowledge assistance. I present the Knowledge Rocks Framework in Chapter 7 and its application in the Security in Process System in Chapter 8. The user studies we performed for the Security in Process System in its original and enhanced version are presented and compared in Chapter 9.

I conclude my thesis and summarize opportunities for future research in Chapter 10. With this thesis I make the following contributions together with my collaborators:



The tree alignment of contour trees is a novelty. Using our heuristic algorithm to quickly compute the alignment of multiple contour trees with a problem-specific similarity metric gives this alignment process flexibility and enhances it to multiple trees.

I introduce a novel layout algorithm for multiple contour trees, that uses both – alignment and individual trees– to achieve a simultaneous, easy-to-interpret visualization of differing topological structures. This is further supported by adjusted interaction possibilities.

We enhance the back-end and front-end of the Fuzzy Contour Trees to adapt our visualization to time-varying scalar fields. The result is a new, holistic visualization of topological structures over time that is more flexible than common approaches like feature tracking.



I define Visitation Graphs, a space saving presentation for large vector field ensembles that is the optimal basis for visitation map calculation and provides a trade off between generation time, calculation time and accuracy. Visitation Graphs allow the interactive exploration of vector field ensembles where naïvely generated visitation maps have immense generation times or are not even available due to lacking information.

To ensure practicality, I provide details on the construction of Visitation Graphs and present an algorithm to quickly approximate visitation maps from our newly defined data structure.



I present the first system for triage analysis in operational technology networks that combines sensor-data visualization and anomaly detection visualization to allow the revision of automatically detected anomalies.

To incorporate knowledge in the Security in Process System, we developed the Knowledge Rocks Framework, providing support for the enhancement of general visualization systems to be knowledge assisted. This framework is derived from and verified using an existing model for knowledge assisted visualization.

We enhance the Security in Process System using the Knowledge Rocks Framework to become knowledge assisted, resulting in further support for triage analysis and the possibility to compare different incidents and collaborate via the system.

We validate the original and the enhanced Security in Process System in detailed user studies, and compare the results from both studies.

In addition, I apply every presented visualization system to different examples (artificial and real-world) and provide details on analysis strategies and usage scenarios. All systems provide different approaches to **compare**, **combine** and **separate** results. They are highlighted in the respective Chapters.

Background: Uncertainty and its Visualization

Uncertainty. A common classification for uncertainty is the distinction between aleatory and epistemic uncertainty [104, 179]. From an application point of view, an uncertainty is epistemic if there is a possibility to reduce them by improved models or data acquisition; aleatory uncertainties on the other hand can not be reduced: they are caused by the random nature of physical events and are modeled by probability theory. In both cases, communicating the uncertainty is crucial to provide the complete picture of the data, without the misleading impressions of certainty. Hence, uncertainty visualization is an important topic and an active field of research.

Uncertainty in data visualization does not only include the visualization of uncertainties that are provided with the data: it can also be introduced in the course of the visualization pipeline [27]. While these uncertainties are important to note, this thesis deals with the visualization of uncertainty, not the uncertainty of visualization.

Uncertainty information associated to data comes in different forms, corresponding to its source. Possible forms and sources are:

- the description of the data as a random variate with a given probability distribution, for example in random fields (see Chapter 5). These frequently arise from in situ pre-processing when summarizing statistics of properties.
- multi valued data resulting from an ensemble of simulations, multiple measurements or time-varying results (Part I),
- model uncertainty arising from imperfect models, including machine learning approaches (Part II), and
- a known range of error originating for example from measuring inaccuracies.

An example for the last point is my work on virtual topography measurements with Andrej Keksel [*101].

In this work, I supported the matching of time series to be able to model measuring inaccuracies for specific surface topography measuring instruments. This is possible since these inaccuracies are aleatory and known. Applying the obtained transfer functions to a topography, the result resembles the measurements of the corresponding instrument, providing information on the practicality of the specific instrument for the measurements at hand.

Like in this example, error ranges and data given as random variate typically represent aleatory uncertainty. Multi valued data can be aleatory or epistemic: if it results from multiple executions of the same experiment, the uncertainty is aleatory. As a result of multiple simulation runs, the variety is likely to result from model uncertainty or epistemic uncertainty of input parameters. Thus, the uncertainty of the multi valued data is epistemic in this case.

Visualization. The importance of uncertainty communication and thus uncertainty visualization has been recognized already in the mid 80s in geosciences [127]. Since then, uncertainty visualization has been an active field of research, gaining importance with the increase of computational power allowing extensive data generation.

On the one hand, uncertainty communication is important since it influences decisions and confidence in the data [46]. Decisions that are made based on visualizations can reach from simple design decisions to life-or-death decisions, for example in medicine or concerning evacuations. Especially in fields with such a crucial influence of uncertainty, supporting the user in reducing the uncertainty in interplay with the visualization system is an important topic (e.g. [157]).

On the other hand, existing uncertainty visualization approaches need to be compared and evaluated: uncertainty representations that are straight forward for scientists might be misleading for laymen or create wrong impressions. For example Padilla et al. found out that common visualizations of possible hurricane tracks lead to both, overestimating and underestimating, the size of the hurricane depending on the chosen visual encoding of the uncertainty [151].

Bonneau et al. give an overview of uncertainty visualization techniques [22] and Potter et al. developed a taxonomy basing on the data and uncertainty dimension [156]. More result-oriented, Gershon classified uncertainty visualization approaches in intrinsic and extrinsic [66]; intrinsic uncertainty representations communicate

the uncertainty via visual properties of the visualized objects, while extrinsic representations are “stand alone” visualizations of the uncertainty using auxiliary objects, for example error bars.

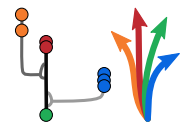
Examples for uncertainty visualization based on ensembles are: showing ensemble members vanishing over time [45], enabling the user to compare single members to the whole ensemble using glyphs [165] and summarizing ensemble members while highlighting outliers and median in Contour/Curve Box Plots [133, 209]. The topology of ensembles in two and three dimensions was determined by Otto et al. in [150] and [149]. Hummel et al. gave a comparative visual analysis for ensembles of time-varying vector fields using a Lagrangian framework [87]. A two dimensional comparative visual analysis was presented by Jarema et al. in [93].

Uncertainty arising from interpolation and prediction of missing measurements was treated using tubes of varying size [21], glyphs and parallel coordinates for MR spectroscopy data [59, 60], flow radar glyphs for time-dependent vector fields with uncertainty given as an interval [82], and using colormapping and line glyphs for uncertain isosurfaces in geosciences [219]. Uncertain predicted multivariate data was visualized by Berger et al. using parallel coordinates and scatter plots [19]. Random fields were treated using fuzzy set theory and volume rendering on trapezoidal possibility distribution [62], and by visualizing iso-surfaces in uncertain scalar fields [153, 155]. An FTLE like method was presented by Schneider et al. [166].

This thesis is concerned with parameter dependency in multi valued data and uncertainties arising from automated data analysis in cyber security. In the following, I present visualization systems that communicate uncertainties in different application backgrounds within these two fields. For each application, specific related work is discussed in the respective chapter.

Part I

Parameter Dependency



Due to growing computational ability, it has become straightforward to investigate the effects of model and parameter uncertainty through *ensemble* simulation. Models are realized multiple times with varying input parameters or settings; each result is a *member* of the ensemble of realizations. An example for this is the weather forecast: A 10% chance of rain is announced if 10% of the evaluated simulations are predicting rain. Also non-virtual experiments are carried out multiple times to get an idea about their uncertainty. An extreme example for this are the experiments at the LHC at CERN: trying to resolve events that arise from particle collisions with low probabilities, the uncertainty of the outcome is high. Thus, 1 billion events per second are recorded over months, producing about one Petabyte of data every second [64]. Since it is prohibitive to store this amount of data –or even transmit it to the surface– *in situ techniques* are employed to analyze and pre-process the data at generation time and store only reduced-size artifacts for further inspection. Similarly, *in situ* techniques are applied in the virtual setting when the size of the generated ensemble requires it or one wants to have insight in the data behavior during generation. Instead of the raw data, images, videos, or other reduced representations are stored. After potential *in situ* pre-processing, the resulting ensembles are analyzed as a whole. Analyzing the similarities and differences between individual ensemble members and sub-ensembles, detection of outliers, as well as the whole ensemble’s development over time gives insights in the dependency of the individual results from input parameters and time. All these analysis tasks can be effectively supported by visualization.

Uncertain Topology of Scalar Field Ensembles

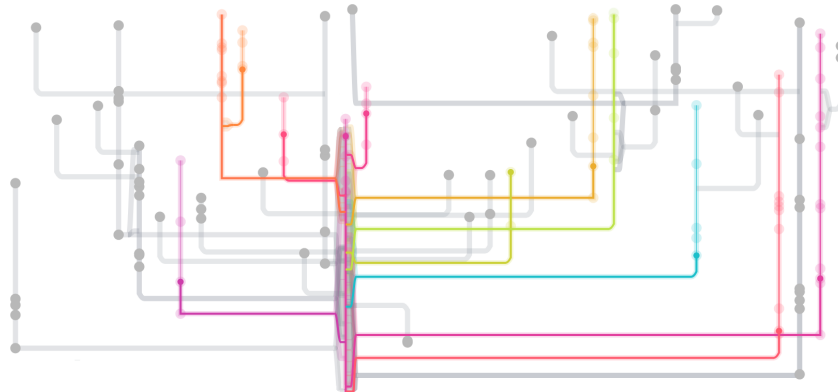


Fig. 3.1.: Fuzzy Contour Trees: Our new visualization of the uncertain topological structure of a scalar field ensemble.

Topology-based methods have a long tradition in the visualization of scalar fields. Founded on mathematical principles, they provide an abstract representation of scalar field structure. Among a variety of methods, the contour tree serves as the well-understood basis for a plethora of techniques, ranging from the straightforward generation of visualization images (e.g. [152]) to clever analysis user interfaces (e.g. [206]).

As modeling and simulation of uncertainty are becoming increasingly prominent aspects of computational science, however, it has proven challenging to adapt topology-based visualization to the resulting novel data modalities. An example for such a challenge is the contour tree visualization of ensemble data sets: the randomized nature of prevalent contour tree layout techniques and their large parameter spaces often result in strongly different representations for very similar scalar fields. Thus, their direct comparison is not sensible.

On the other hand, the meaningful simultaneous visualization of ensemble contour trees is very promising. The encoded topological features provide profound information about the ensemble's behavior. Representing this information in a comprehensive, sensible manner, users are for example able to:

- Compare** ensemble members to identify similarities and differences in their topological structure. An example is the identification of scalar values that induce topological changes in contours in some or all of the ensemble members.
- Combine** ensemble members with common topological segmentation or threshold values. Examples are the assessment of the ability of iso-contours to represent the whole ensemble, or the question which critical points occur in all members.
- Separate** groups of ensemble members with similar behavior and identify outliers. For example members that contain a specific branch can be determined.

By analyzing the overall ensemble without loosing track of individual members, parameter settings of the members can be related to their behavior.

In collaboration with Florian Wetzels, Dr. Jonas Lukasczyk, Prof.Dr. Gunther H. Weber and Prof.Dr. Christoph Garth, I developed *Fuzzy Contour Trees*, the first visualization system that allows the joint visualization of many contour trees. It was published and presented at EuroVis 2020 [*120].

To generate a Fuzzy Contour Tree from multiple member contour trees, a matching between them is required. Based on the application of tree alignments to contour trees –developed by Florian Wetzels, Jonas Lukasczyk and Christoph Garth– we identify common branches across multiple contour trees. Commonality is identified through a semantically meaningful similarity metric that can be chosen freely, providing a high flexibility of our approach. This back end of Fuzzy Contour Trees is presented in the first part of this Chapter (Section 3.2).

Using the resulting alignment, I developed in collaboration with Christoph Garth and Gunther H. Weber an algorithm to lay out common branches for all member contour trees identically. Determining the layout of individual contour trees based on the layout of their alignment (a super tree), we ensure that common branches are rendered at a common position. Hence, superposition of the different member contour trees is meaningful. With several additional layout improvements that I developed together with Frederike Gartzky, the result yields a coherent, easy-to-interpret representation of multiple contour trees of an ensemble at once. The layout algorithm with its optimizations, interaction possibilities and challenges is presented in Section 3.3. Linking between the Fuzzy Contour Tree and individual ensemble members highlights their dependency

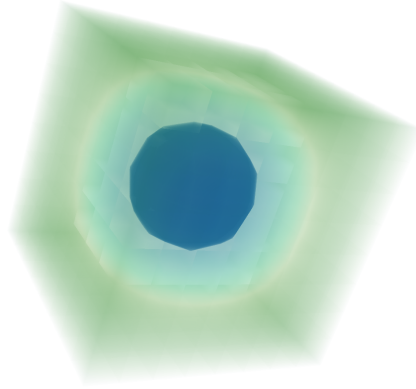


Fig. 3.2.: In this **three-dimensional scalar field rendering**, colors indicate different spans of isovalues. The level sets of specific isovalues are visible at color changes.

3.1 Review of Topology-Based Ensemble Visualization

3.1.1 Contour Trees

Contour lines have been used to visualize scalar fields for almost 500 years [136] e.g. to depict lines of equal height on maps. The idea to draw lines and surfaces along areas of constant values is mathematically defined as the visualization of *level sets*.

Considering a scalar field $s : \mathbb{M} \rightarrow \mathbb{R}$ defined over a manifold $\mathbb{M} \subseteq \mathbb{R}^d$, the level set of for an isovalue $c \in s(\mathbb{M})$ is given as

$$s^{-1}(c) = \{p \in \mathbb{M} | s(p) = c\}.$$

An example for surfaces representing level sets is shown in Figure 3.2. The use of transparency in this visualization allows to show the structure of the scalar field inside the cube. However, isosurface visualization often suffers from occlusion since level sets often include each other or overlap in their two-dimensional projection: for example the behavior of the scalar field inside the blue ball at the center of Figure 3.2 is not visible.

Contour trees (for functions on general, potentially non-flat spaces: reeb graphs) visualize the evolution of connected components of the level sets (“contours”). Two points $x, y \in \mathbb{M}$ with $s(x) = s(y)$ are part of the same connected component if there exists a continuous function $p : [0, 1] \rightarrow \mathbb{M}$ with $p[0] = x$, $p[1] = y$ and $\forall z \in [0, 1] : s(p(z)) = s(x)$. That is x can be reached from y via a continuous path that does not leave the level set.

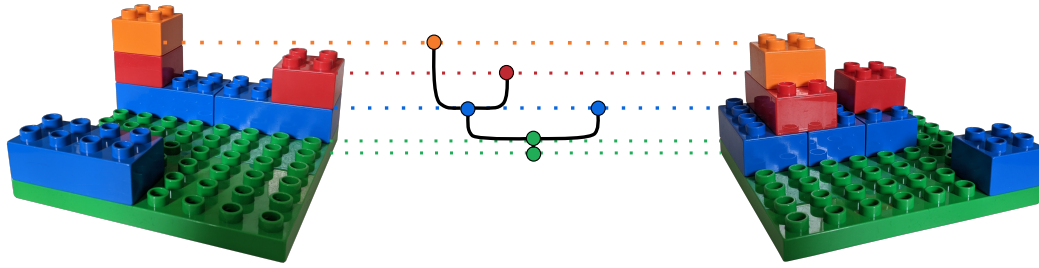


Fig. 3.3.: A **contour tree** representing the (simplified) topological structure of both given height fields.

As the isovalue c changes, the connected components can appear, vanish, split or merge. This evolution of connected components is tracked by the contour tree. As an abstraction of the topological structure, the field corresponding to a contour tree is not unique. An example for a contour tree with two corresponding height fields is given in Figure 3.3. The height value in the contour tree corresponds to the chosen isovalue. Each edge corresponds to one connected component. So, looking at a specific isovalue c (corresponding to a certain height in the tree), the number of edges represents the number of connected components of the level sets $s^{-1}(c)$. Leaves in the contour tree represent emergence or disappearance of a connected component and interior vertices represent merging and splitting.

Changes in the connected components of level sets only occur at critical points. This fact can be used to define and select meaningful level sets [12]. Also, contour trees have been used to define features of interest [26] or provide user interfaces [31, 102, 207]. By pruning contour trees, topological simplification of the corresponding scalar field is achieved [31, 43] and two scalar fields can be compared by direct comparison of their contour trees [167]. Generalizations of contour trees to multidimensional data have been presented by Carr and Duke [32].

For further reading, especially about the efficient computation of reeb graphs, see for example the work by Doraiswamy et al., Carr et al. and Hajj et al. [33, 48, 76].

3.1.2 Visualization of Contour Trees

In contrast to general tree visualization, contour tree layouts are required to fulfill different requirements. Edges in contour trees represent connected components of level sets and thus depend on isovalues. Hence, in a contour tree visualization, the y-coordinates of both their ends are fixed. In addition, further prerequisites are imposed by requirements to consistency and pruning. As a result, common tree or graph layout algorithms are not applicable to contour trees.

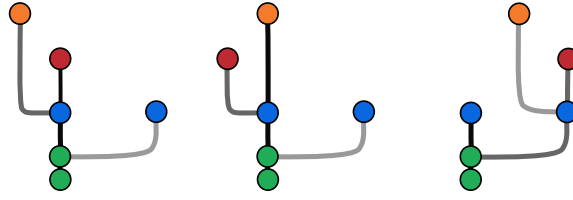


Fig. 3.4.: Different **branch decomposition and layout options** for the contour tree in Figure 3.3 on the facing page. Branches are colored in grey levels.

Heine et al. provide a detailed overview over layout strategies for contour trees and propose a new layout method [80]. As they point out, the center piece of contour tree layout strategies is the *branch decomposition* (Figure 3.4). The unrooted contour tree is decomposed in branches with parent/child relations using a suitable criterion. Often times, this criterion is the edge persistence, that is the difference between the highest and the lowest isovalue reached by the edge; this results in the longest edge being the main branch of the contour tree.

To minimize edge crossings in the contour tree layout, common layout strategies rely to a certain extent on randomization since finding an optimal layout is a NP-hard problem. An example is the orthogonal layout strategy proposed by Heine et al. for crossing minimization [80]. It consists of four phases:

In the first phase, the layout problem is *simplified*. After the branch decomposition, branches that can be drawn without any edge crossing are merged in branch groups.

In the following *permutation* phase, the branch groups are randomly permuted and the algorithm tries to find optimal horizontal positions (in terms of a minimum weighted number of edge crossings) using a random walk and simulated annealing: each branch gets assigned a random vertical slot in the layout. In each step, a branch group is re-inserted at a random slot and the resulting order of branch groups is rated. If the new order is rated better than the previous ordering, it is kept. Else, if the rating of the new order deteriorates by Δ , the new order is kept with probability $\exp(-\frac{i\Delta}{\tau})$ with i the number of iteration and τ a user defined parameter. The best ordering is stored and the algorithm terminates if after a given maximum number of iterations no further improvements have been found.

The *order* phase extends the ordering of branch groups to an ordering of branches using the silhouette idea of the Reingold-Tilford layout on a directed acyclic graph [160], derived from the contour tree. The final horizontal positions are then assigned in the *position* phase similar to the position phase of dot [65], considering the vertical extent of branches and possible overlaps of vertical slots.

While the randomization of branch positions allows to find good layouts without calculating all possibilities, it implicates that two layouts of similar contour trees –or even the same one– are possibly very different. Hence, comparing contour trees is impossible using such layout strategies.

3.1.3 Topology-Based Ensemble Visualization

Applying topology-based visualization to multiple scalar fields at once has several major use cases. In ensemble analysis, an understanding of commonalities and differences between ensemble members is sought [203], while the study of time-dependent scalar fields mostly aims to identify feature evolution over time [26]. In both cases, an important problem is to establish feature correspondence by topological means. A common approach is to use branches or sub-trees of contour trees to characterize regions that are then examined for correspondence using overlap measures; however, this does not take the contour tree structure into account. An example is the comparison of two scalar fields based on contours obtained from the contour tree by Schneider et al. [167]. Similarly, Lukasczyk et al. uses merge tree segmentations to compute the correlation between features [125]. Space-filling structures in turbulent flows are tracked by Schnorr et al. using the volume overlap of three-dimensional Morse-Smale cells, which serve as input to a maximum-weight, maximal matching [170].

Instead of considering the spatial overlap of topologically-characterized regions of scalar fields, an other class of methods focuses primarily on correspondence directly from a graph-centric perspective. For example, Saikia et al. [164] compare all sub-trees of two merge trees against each other to find repeating structures, and Thomas and Natarajan [193] adopt a similar approach to identify symmetries in scalar fields.

The visualization of scalar field ensembles using contour trees as visual representation involves the comparison of trees as well as the visualization of uncertain tree structure. For example the information in, and differences between, multiple trees was visualized by Schulz et al. through an edge-bundled visualization of multiple samples from a probabilistic graph model [171]. Location and sub-tree structure uncertainty of two different graphs were visualized by Lee et al. [110], and Shu et al. discuss EnsembleGraphs to visualize hierarchical clustering across an ensemble [177].

Contour trees of uncertain scalar fields were considered by Kraus [106]. Here, two contour trees of morphologically filtered versions of an uncertain volume data

set represent the range of uncertainty, visualized by combining both trees in one image. Günther et al. [74] also use two realizations of an uncertain scalar field that represent estimations of the support of the probability density function of the input data. They characterize mandatory critical points in the given range of realizations and provide mandatory merge and split trees.

Contour tree-based uncertainty visualization as proposed by Wu et al. [214] includes a layout algorithm for contour trees. Similar to the idea by Heine et al. [80], they assign slots to branches. The same authors visualize the mean contour tree obtained from the pointwise ensemble mean, with uncertainty added from contour differences between individual members. While this contour tree summarizes information about the whole ensemble, there is no link between the mean contour tree and the member contour trees. Their individual information is lost in the ensemble visualization and outliers are hidden since they are “overruled” by the other members.

To keep the individual information of member contour trees and still be able to create a single visualization for the ensemble, the identification of nodes reflecting similar areas in the member fields is necessary. In the following, I describe the major classes of techniques that are used for distance measurement and merging of graph-based topological descriptors (e.g. contour trees).

The general problem of finding a distance between (rooted) trees arises in different fields of computer science, such as computational biology [176], AI [105] and code compilation [83]. Various types of edit distances, based on defining a cost function for edit operations in trees, have been applied to solve this problem, with the tree edit distance [191] being the most general and complex approach. An overview is given by Bille [20]. Tree alignments, a computationally cheaper alternative, were introduced by Jiang et al. [96].

Recently, different types of edit distances have been applied to merge trees and other graph based descriptors representing the topology of a scalar field. Saikia et al. [164] applied the 1-degree edit distance to branch decomposition trees of merge trees to find self similarities in scalar fields. Sridharamurthy et al. used the constrained edit distance [221] on merge trees for feature tracking in time-dependent data [183]. Beketayev et al. [16] propose a method to compare merge trees based on the minimum edit distance between all possible branch decompositions of the two compared trees. Rieck et al. use the edit distance for ordered trees on persistence hierarchies [161]. Moreover, many metrics other than edit distances have been proposed for merge trees, often obtained by restricting a metric on the more general Reeb graph [13, 14, 34, 138, 178]. Yan et al. introduced a metric between labeled merge trees, allowing the definition of an average of several merge trees [216].

Contour trees pose more challenges than merge trees when searching distance metrics or matching algorithms. They are in general more complex data structures with potentially high variance for small changes in the considered scalar field. As recently shown by Hristov et al., also branch decomposition poses additional challenges for contour trees [85]. Applying their method to contour trees, Saikia et al. [164] describe similar problems.

Our work on Fuzzy Contour Trees is located in this gap concerning the application of edit distances and general merging of contour trees, and the application of contour trees to ensembles without losing the information of individual members. We apply alignments to contour trees; apart from being easier to compute, they exhibit some properties which make them a good fit for our purpose: The distance between single contour trees is not of interest but a matching of their nodes is required to achieve a common layout. To do so, the resulting matching needs to incorporate all paths and features of the single contour trees. This makes the edit distance the preferred approach for our purpose in terms of the tree alignment, resulting in a super-tree with the required properties. Those properties are explained in Section 3.2 as well as our method to apply the alignment to an ensemble of more than two unrooted, unordered contour trees.

3.2 Tree Alignment of Contour Trees

We aim to devise a combined representation of multiple contour trees that respects and leverages similarities among the trees and the scalar fields they represent, to facilitate common, topology-based analytical tasks. A central problem of this is the identification of such similarities. This can be accomplished –on a tree level– by a matching between the nodes and arcs of all individual contour trees, such that matched nodes and arcs correspond to similar structures in the scalar fields. A good way to find such a matching is using tree edit distances, which induce a mapping of nodes in the compared trees [20]. Like this, the trees become minimally different w.r.t. edit distance.

In brief, edit distance between two *labeled* trees (that is the tree nodes have a label identifying a matching between the nodes of both trees) measures the minimum number of operations required to transform one tree into the other. Typically these operations are:

- inserting** a node in one tree that does exist in the other tree,
- deleting** a node in one tree that does not exist in the other tree and

relabeling a node.

Operations can carry arbitrary cost, and a cost-minimal sequence of edit operations is sought. An edit sequence S for two trees T_1 and T_2 induces a mapping of a subset of their vertices $M_S \subset V(T_1) \times V(T_2)$ where for all $(v_1, w_1), (v_2, w_2) \in M_S$

- $v_1 = v_2$ if and only if $w_1 = w_2$, and
- v_1 is an ancestor of v_2 if and only if w_1 is an ancestor of w_2 .

Given two rooted trees T_1 and T_2 for which an ordering is specified for the children of each vertex (*ordered trees*), the edit distance $\delta(T_1, T_2)$ can be computed in time $\mathcal{O}(|T_1| \cdot |T_2| \cdot |L_1| \cdot |L_2|)$ using dynamic programming, where L_1 and L_2 are the depths of the trees [191]. If the two rooted trees are unordered, the problem of computing the value of $\delta(T_1, T_2)$ is known to be NP-hard [20].

Contour trees are unordered, unrooted trees, thus the general edit distance is too costly for our purpose. However, many restricted variants of the edit distance have been introduced [20], which can be computed on unrooted trees and at lower cost. From these, we utilize *tree alignments* and the corresponding *tree alignment distance*.

A tree alignment \mathcal{A} of trees T_1, \dots, T_n is a super-tree of the aligned trees, i.e. it contains each aligned tree as a sub-tree. In general, \mathcal{A} is not unique and can be computed from each individual tree through sequences of insert operations and node relabelings. A *minimal tree alignment* minimizes a cost function over the edit sequences that yield \mathcal{A} from each T_i , thus intuitively providing a “small” alignment that captures the similarity between the individual trees.

In comparison to general edit mappings, whose computation is NP-hard, minimal alignments can be found in quadratic time in the number of nodes for (arbitrarily rooted) contour trees. Furthermore, an important property towards a joint layout of contour trees is the *path property*: all paths in the individual trees map to paths in the super-tree. A detailed description and comparison of these concepts can be found in the survey by Bille [20].

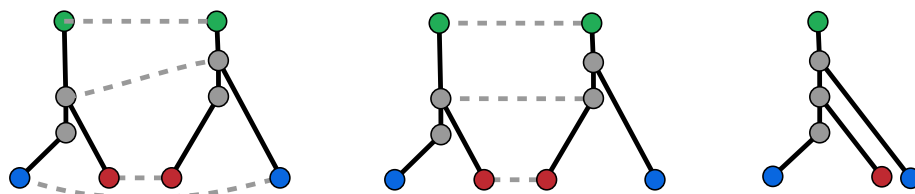


Fig. 3.5.: Differences between alignment and edit distance: the intuitive mapping (left) can not be achieved by the alignment but it requires an edit distance mapping. The minimal alignment (center) induces the mapping on the right.

3.2.1 Minimal Contour Tree Alignment

An alignment of two trees T_1 and T_2 is obtained by *first* inserting nodes labeled with a blank symbol λ into T_1 and T_2 , making them isomorphic. Let T'_1, T'_2 be the resulting trees and T_{align} be the unlabeled tree isomorphic to both. Labeling a node $v \in V(\mathcal{A})$ with $l(v) := (l(v_1), l(v_2))$, where v_1 and v_2 are the nodes in T'_1 and T'_2 corresponding to v , and l is the labeling, gives the alignment \mathcal{A} .

The alignment label $l(v)$ represents an edit operation, and is associated with a cost $\gamma(l(v_1), l(v_2))$, where γ is an arbitrary metric. The overall cost of \mathcal{A} is then

$$\gamma(\mathcal{A}) = \sum_{v \in V(\mathcal{A})} \gamma(l(v)),$$

allowing to define the alignment distance as the minimal cost

$$\delta_{\text{align}}(T_1, T_2) = \min\{\gamma(\mathcal{A}) \mid \mathcal{A} \text{ is alignment for } T_1, T_2\}.$$

Each \mathcal{A} minimizing $\gamma(\mathcal{A})$ is a minimal tree alignment for the chosen metric.

The minimal alignment of trees corresponds to a restricted edit distance, where all insertions are performed before all deletions. This yields the super-tree property, and nodes labeled without λ represent the induced matching.

Differences between Alignment and Edit Distance. To illustrate the behavior of tree alignments, I highlight a number of differences between tree alignments and edit mappings. From an edit distance mapping between T_1 and T_2 , one can construct a tree of the mapped nodes in a natural way (following from the mapping properties). This tree will always be a sub-tree of T_1 and T_2 .

Given the cost function

$$\begin{aligned} \gamma(\lambda, l) &= \gamma(l, \lambda) = 1, \\ \gamma(l_1, l_2) &= \begin{cases} 0 & \text{if } l_1 = l_2 \\ 2 & \text{otherwise} \end{cases} \end{aligned}$$

the minimal alignment from T_1 to T_2 will be the smallest common super-tree, and the sub-tree induced by the minimal edit sequence will be the largest common sub-tree [20]. Therefore, alignment mappings are not able to match certain corresponding structures; consider e.g. the alignment and edit mapping in Figure 3.5 on the previous page: on the left, an intuitive mapping is given that requires an

edit distance mapping, as the lower gray node must be deleted as a parent of the blue node and a new gray node must be inserted as parent of the red node. In an alignment though, insertions must occur *before* deletions. Hence, the alignment procedure will never result in this matching on the left. The less intuitive result of the alignment is given at the center and on the right; matching the blue nodes is impossible since it would result in a cycle.

However, the super-tree property provides substantial advantages. First, it allows the construction of a heuristic for aligning more than two trees (cf. Section 3.2.2). Furthermore, a super-tree of all contour trees contains all features (critical points) of the original fields. In contrast, an edit mapping only induces a sub-tree. A further important advantage of alignments is reduced computational complexity: for two unordered trees with bounded degree, the alignment can be computed in time $\mathcal{O}(|T_1| \cdot |T_2|)$ in contrast to the NP-hard edit distance problem [20]. This assumption is fulfilled for contour trees in most practical settings (e.g. in the strongly prevalent piecewise linear case).

3.2.2 Alignment Heuristics

We extend the minimal alignments introduced above from two trees to n trees as follows. Given n scalar fields, the alignment of the corresponding contour trees can be used as a representation of the topology of the ensemble. In general, the problem of aligning n trees is again known to be NP-hard, even for bounded degree or ordered trees, since it is a generalization of the multiple sequence alignment [204]. Thus, direct computation is not feasible. Furthermore, the alignment procedure requires rooted trees, whereas contour trees are unrooted. To address both problems, we adopt the following interlocking heuristics:

Sequential Alignment of Multiple Trees. Let \mathcal{A}_2 be the minimal alignment of T_1 and T_2 , and define \mathcal{A}_{k+1} as the minimal alignment of T_{k+1} and \mathcal{A}_k . The final matching is the one induced by \mathcal{A}_n .

In this manner, we construct an alignment of n trees sequentially. This alignment will in general not be a minimal alignment. However, \mathcal{A}_2 contains all features of T_1 and T_2 . Aligning a third tree T_3 which has features similar to T_1 but not to T_2 , with \mathcal{A}_2 , the resulting alignment \mathcal{A}_3 will still match them, since the features are present in \mathcal{A}_2 and T_3 . For example, consider the two trees in Figure 3.5 on page 21. The blue and red nodes are swapped. If further trees with this swap are aligned,

there will likely be two blue and two red nodes in the alignment. Our experiments (cf. Section 3.4 on page 37) indicate that this heuristic works well in practice and is cheap to compute.

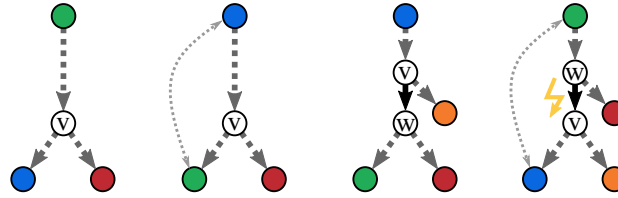


Fig. 3.6.: Consistent root and path properties: (left to right) for a given alignment, a different root is chosen and another tree is added to the alignment. This alignment results in violation of the ancestor property for node v if the the root is switched back.

Rooting Contour Trees. To align two unrooted contour trees, it appears possible to minimize alignment over all possible choices of roots. For the sequential alignment, this can however lead to problems; in Figure 3.6, the edit mapping property is violated after aligning with respect to different roots. This problem does not arise if the root of the alignment is kept consistent. Thus, in each step of the sequential alignment, the alignment node corresponding to the previously chosen roots has to be chosen as the root of the new alignment as well. In contrast, the root of T_{k+1} can be chosen freely to obtain an optimal result.

3.2.3 Cost Metrics

The cost of edit operations that induce the minimal alignment can be chosen as an arbitrary metric, providing flexibility in steering minimal alignments towards matching nodes that are semantically related. For example, nodes can be labeled by scalar value, and the difference between the values of two nodes can be chosen as cost. A similar construction, independent of the absolute scalar value, can be obtained by labeling nodes in a rooted tree with the difference in scalar value to their origin, i.e. with the persistence of the unique edge pointed to this node. Again, the metric is the difference of the two matched leaves. One could also use the area corresponding to this edge in the field, or the sum or product of several of these quantities, depending on application needs. Following Sakia et al. [164], we call the size of the edge segmentation *volume*, independent of the dimension, and the product of *volume* and *persistence metric* the *combined metric*. In their use case, the combined metric performed best. A purely combinatorial matching is possible by defining fixed costs per edit operation type, but this appears less useful in the

application scenarios we envision here, since we are aiming on topological similarity of matched features and proper labeling of the resulting Fuzzy Contour Tree.

To be able to use an alignment as an input tree for the next alignment, a meaningful way to combine labels of the form $(l(v_1), l(v_2))$ into a single label after each alignment step needs to be chosen. For example, for scalar value labels, the average of $l(v_1)$ and $l(v_2)$ can be chosen as the new label. Similar constructions can be used for the other examples discussed above.

Importantly, to preserve the semantics of the individual contour trees in an alignment, we penalize the matching of nodes of different critical point type (minimum, maximum, saddle) by choosing prohibitively large cost for such relabelings. Hence, we ensure that it is always cheaper to insert a new node than to match critical points of different types.

3.2.4 Algorithm

Algorithm 1: Heuristic for minimal alignment of n contour trees

Let \mathcal{A}_{min} be some alignment tree with infinite cost

```

foreach leaf  $r_1$  of  $T_1$  do
  Let  $T_1^{r_1}$  be  $T_1$  rooted in  $r_1$ 
   $\mathcal{A} = T_1^{r_1}$ 
  for  $i = 2 \dots n$  do
    Let  $\mathcal{A}'_{min}$  be some alignment tree with infinite cost
    foreach leaf  $r_i$  of  $T_i$  do
      Let  $T_i^{r_i}$  be  $T_i$  rooted in  $r_i$ 
       $\mathcal{A}' = \text{align}(\mathcal{A}, T_i^{r_i})$ 
      if  $c(\mathcal{A}') < c(\mathcal{A}'_{min})$  then
        |  $\mathcal{A}'_{min} = \mathcal{A}'$ 
      if  $c(\mathcal{A}'_{min}) < c(\mathcal{A})$  then
        |  $\mathcal{A} = \mathcal{A}'_{min}$ 
    if  $c(\mathcal{A}) < c(\mathcal{A}_{min})$  then
      |  $\mathcal{A}_{min} = \mathcal{A}$ 

```

The overall algorithm to approximate the minimal alignment for n contour trees T_1, \dots, T_n with cost metric c is shown in Algorithm 1. Allowing arbitrary choices for the cost metric, it is very flexible and can be adapted to the needs of a particular application domain.

The runtime of the above algorithm is in $\mathcal{O}(n^2 \cdot |V_{max}|^4)$ for n trees, where $|V_{max}|$ is the number of nodes of the largest tree. Because this is still expensive for large trees, and it is not sensible to lay out contour trees with hundreds of nodes, we apply contour tree simplification (e.g [31]) before alignment. This results in very good computation times for trees with several hundreds of nodes, as given in Table 3.1 on page 38 for the examples discussed in Section 3.4.

The given algorithm is at heart a randomized algorithm; finding an ordering of trees to ensure optimal alignment is a NP-hard problem. Thus, we randomly permute the input ordering of trees, as is done in many other algorithms that would otherwise have to employ exhaustive combinatorial search. In practice, to increase repeatability, the random ordering is computed using a fixed chosen seed. In our experiments, we have found that while alignments differ, the quality of the resulting layouts is largely independent of the chosen seed. Figure 3.13 on page 35 shows layouts resulting from two different seeds for the same set of contour trees.

3.2.5 Properties of the Contour Tree Alignment

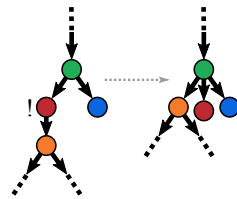


Fig. 3.7.: The red inner node representing an extremum is turned into a leaf.

The output of our algorithm is an alignment of the n contour trees, where each T_i is rooted in a chosen leaf, and all roots are matched to the root of the alignment. \mathcal{A} fulfills a set of properties that are important for the layout algorithm:

- \mathcal{A} is a super-tree, therefore all inner nodes of the individual trees are matched to inner nodes of the alignment and all leaves of the alignment represent leaves of the individual trees.
- The alignment preserves the node type, i.e. the alignment nodes also have a specific type (minimum, maximum or saddle).
- All paths in individual contour trees which start at the chosen root are matched to sub-paths in the alignment (path property).

Some properties of the alignment complicate laying out the Fuzzy Contour Tree:

- In contrast to contour trees, an inner node of the alignment can be a minimum or maximum. For visualization purposes, these inner extrema nodes can

be turned into leaf nodes by attaching their children to their parent node. (cf. Figure 3.7 on the preceding page and Section 3.3.1).

- Matched leaves from different contour trees are not necessarily connected to a single matched saddle. However, the path property ensures that different saddles will be either on the parent branch of the leaf in the alignment or in its sub-tree.
- For a saddle node in the alignment, that matches saddle nodes from the member trees, there is not necessarily a single leaf node matching leaves from exactly the same member trees.

3.3 Fuzzy Contour Trees

Based on the alignment described in the previous section, we defined a layout algorithm that allows an intuitive joint depiction of multiple contour trees in a sensible manner –the *Fuzzy Contour Tree*.

In order to achieve a high recognition factor for the Fuzzy Contour Tree, we use the well-established and often-used orthogonal layout [80] as a basis for our algorithm. In this layout, branches are drawn as vertical lines. They are connected by saddles, which are drawn as horizontal lines rather than points. Finding an orthogonal layout for the alignment (and thus for all aligned contour trees) is done in analogy to finding a layout for (single) contour trees. First, a branch decomposition is recursively established, then the resulting branches are assigned horizontal positions, with the vertical positions of the nodes given by their isovalues. Matched nodes in the individual contour (sub-)trees of the alignment are assigned equal positions. They are then combined, and further layout improvements for the resulting Fuzzy Contour Tree are performed to further increase visual clarity. In the following, I contrast our approach to the layout for single contour trees and present the layout improvements in detail.

3.3.1 Branch Decomposition of the Alignment

A key ingredient in contour tree layout is the branch decomposition. To identify a branch decomposition of a contour tree, first, a root and a main branch are selected. From saddles in this main branch, further branches can be identified recursively until the entire contour tree is decomposed. In case of a single contour tree, the leaf with

minimum isovalue is chosen as root, and the main branch is chosen as the monotone increasing path with maximal persistence starting in this root. Considering multiple contour trees at a time, these properties may vary between individual contour trees. Thus, the choice of root and main branch is more complicated.

The alignment provides a dedicated root node. This root is guaranteed to exist in all individual contour trees and ensures the path property of the alignment (see Section 3.2.2 on page 23 and Figure 3.6 on page 24). In the process of branch decomposition, this root might turn out as the leaf with maximal instead of minimal isovalue in the main branch of individual contour trees. In this case, the minimum of the main branch is considered as root.

Starting in the chosen root node, a main branch is chosen by considering both – alignment and individual contour trees– as follows: paths in the alignment from the root node to each leaf are initially considered as candidates for the main branch. Note that paths that are monotone in one or more individual trees are not necessarily monotone in the alignment, due to insertion of nodes and averaging of labels (isovalues). Separately for increasing and decreasing directions, each candidate path in the alignment is then considered in each individual tree, and counted if it exists in this tree and is monotone, which gives its *path frequency* F . This frequency and the path persistence P are then used to obtain a rating R for candidate paths.

$$R := P_{\%} \cdot w_{\text{pers}} + F_{\%} \cdot w_{\text{freq}} \text{ where}$$

$$P_{\%} := \frac{P \cdot 100}{I_{\text{max}} - I_{\text{min}}} \text{ and } F_{\%} := \frac{F \cdot 100}{n}$$

Here, $P_{\%}$ is the percentage of the path persistence relative to the isovalue range $[I_{\text{min}}, I_{\text{max}}]$ of the alignment, $F_{\%}$ is the percentage of the n contour trees that contain the considered path, and $w_{\text{pers}} + w_{\text{freq}} = 1$ are user-chosen weights for persistence and frequency. These weights are further discussed in Section 3.3.4 on page 34.

Choosing the path with the highest rating R as the main branch and proceeding recursively for each sub-branch (i.e. saddle) of the main branch yields a branch decomposition of the alignment. A corner case occurs if no contour tree contains a path from the currently considered saddle to any leaf. The frequency of the branch is then considered zero, and the rating is based only on the path persistence in the alignment.

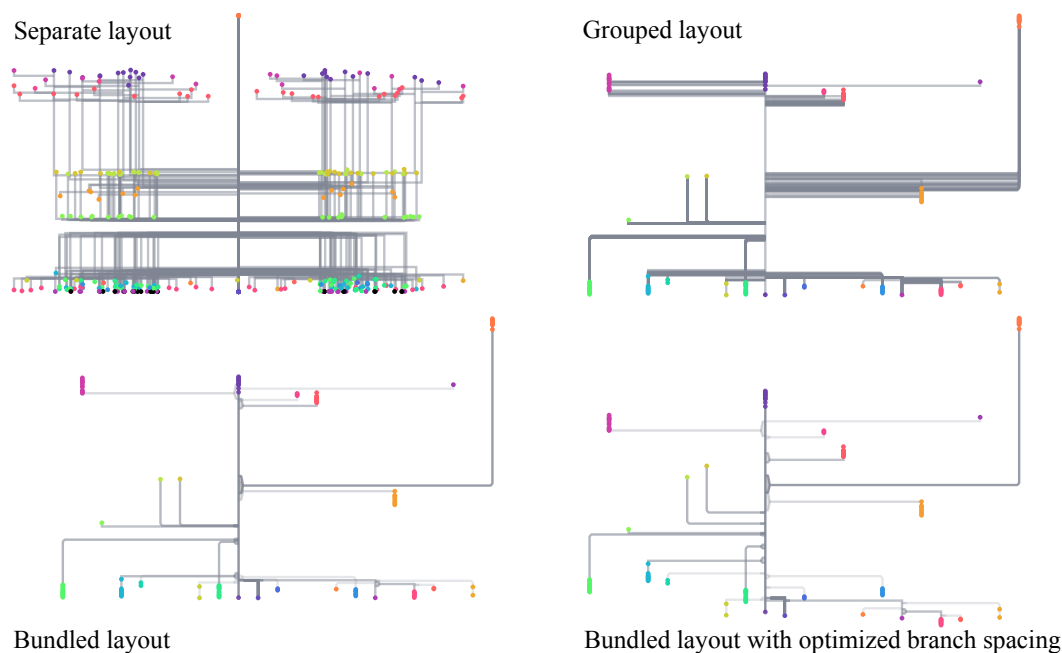


Fig. 3.8.: An illustration of the **Fuzzy Contour Tree layout**: *Separate layout* of multiple contour trees yields a cluttered representation, while *grouped* and *bundled* layout position aligned branches jointly. To better leverage vertical space, saddles can be shifted using *optimized branch spacing*. All steps are treated in detail in the following.

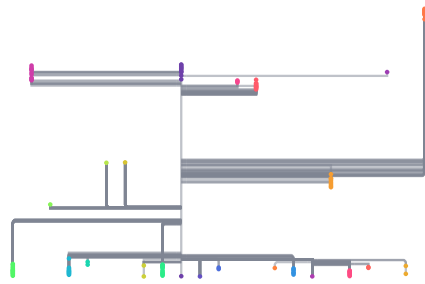
3.3.2 Layout Algorithm

After a branch decomposition for the alignment is established, many known layout algorithms for contour trees could be employed. To obtain a suitable layout for the Fuzzy Contour Tree representing the combination of all individual contour trees, additional information from the individual trees needs to be taken into account when optimizing layout clarity, e.g. by minimizing crossings. To do so, we incorporate the isovalues of nodes from individual contour trees into the layout, resulting in value ranges (as opposed to individual isovalues) for leaves and saddles. Further influences of individual contour trees on the bundled layout are described after its introduction.

As a basic layout strategy, we adapt the (partly randomized) method proposed by Heine et al. in their permutation phase [80]: we attempt to find an ordering of branch groups that minimizes a weighted number of edge crossings. Instead of branch persistence, we weight crossing by the rating R obtained during branch decomposition. Thus, branches that have been chosen as main branches for the entire alignment or sub-trees are less likely to be crossed in the resulting ordering. The optimum ordering is sought as proposed by Heine et al. using a combination

of random walk and simulated annealing. While this approach does not ensure an optimal layout, it gives very good results in practice (cf. Section 3.4 on page 37). Furthermore, the algorithm’s non-deterministic nature may yield different layouts given similar or identical input; Yet, since it is applied to the alignment, a super-tree of all member contour trees, the derived layouts for the individual contour trees always match. Calculating randomized layouts of the individual contour trees, this is not the case.

In our setting, all branches are considered as individual branch groups. This is a natural choice, since the decomposition of the alignment into branch groups, taking multiple isovalues per node into account, tends to result in small branch groups, often containing only a single branch. The resulting order of branches is translated directly into horizontal coordinates for the layout, such that each branch occupies one vertical slice of the overall layout.



Grouped Layout. The horizontal coordinates obtained in the alignment layout can be propagated to the individual contour trees via the node matching from the alignment. Thus, across all contour trees, matched nodes are assigned identical horizontal positions and keep the vertical position according to the isovalue. Superimposing all individual contour trees with the assigned mutual layout results in the *grouped layout*.

While this layout presents a significant improvement over separate layout of individual trees with superposition (“separate layout” in Figure 3.8 on the previous page), visual clutter is still an issue and can be disruptive.



Bundled Layout. To reduce visual clutter, we further abstract the grouped layout through edge bundling. On the basis of the grouped layout, the *bundled layout* bundles all edges of a branch group and assigns an opacity to edges and nodes based on the rating R of the branch. Branch edges are bundled close to their origin to the mean vertical position of the group’s saddles. To

further simplify the representation, we only draw the edges of a branch group originating at the respective maximum and minimum saddle isovalues. The inbetween

area is filled with appropriate opacity to accentuate its affiliation. Challenges arising in this visualization are discussed in Section 3.3.5 on page 35.

Optimized Vertical Branch Spacing. In many cases, ensemble members will have a similar topological structure, resulting in a strong resemblance of their contour trees in the common layout. This may result in clustered branch origins in the Fuzzy Contour Tree. To disambiguate in these cases, we propose to shift branches vertically to better leverage available vertical space. Although in this case, the vertical node position no longer indicates the isovalue, we preserve the vertical ordering of branches. Furthermore, the saddle isovalue ranges of two branches left and right of the parent branch overlap only if they do so in the original tree and, given sufficient vertical space, the vertical distribution of branches on each parent branch adhere to the original distribution as much as possible.

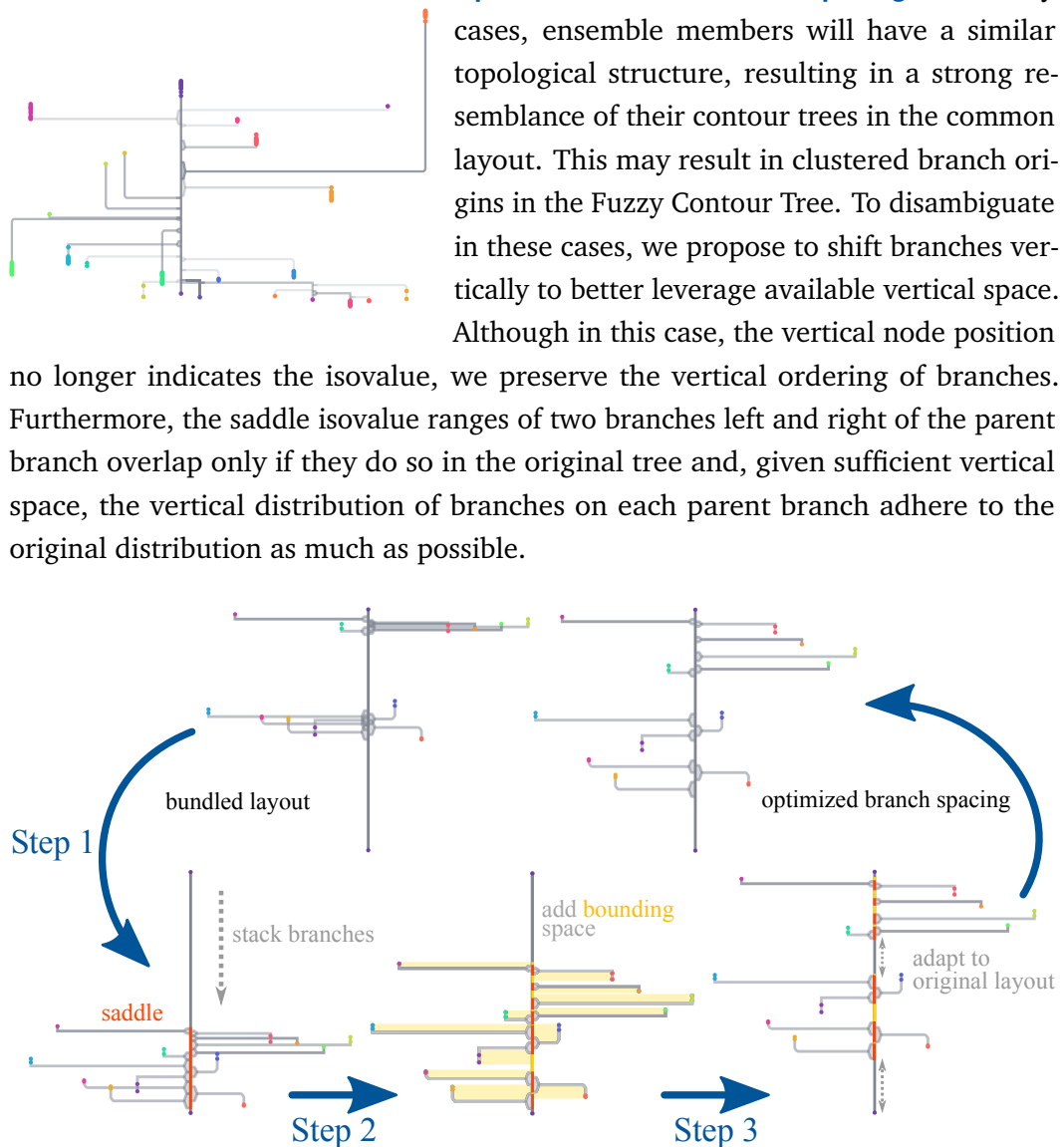


Fig. 3.9.: Branch spacing optimization: (bottom left to right) **Step 1** all saddles are stacked. **Step 2** Spaces based on bounding boxes are added. **Step 3** The branch distribution is adapted to the original layout.

The shifting procedure is performed across all sub-trees of the Fuzzy Contour Tree in a bottom-up manner, beginning with the branches farthest from the chosen main branch. Available space on a sub-tree’s main branch is filled in three steps, with different types of spaces considered in each step (cf. Figure 3.9 for an illustration):

- Step 1** All saddles are stacked in correct order without space in between. Overlaps of saddles left and right of the main branch are maintained. The occupied vertical space is marked as “saddle”.
- Step 2** Based on the bounding box of the sub-tree’s main branch, “bounding” spaces are added above and below every “saddle” space, if the current space on the respective side of the main branch is smaller than the bounding box (plus a user defined threshold).
- Step 3** The original space above and below every child branch on the sub-tree’s main branch is compared to the current spacing. Space is added to obtain a distribution of the child branches similar to the original layout.

After each step, the amount of occupied vertical space relative to the available height is checked. If it exceeds the available height, the spaces added in the previous step are “compressed” by scaling all vertical heights down such that the maximum available height kept; all further steps are omitted. If this occurs after the first step, this means that an overlap of the isovalue ranges cannot be avoided. After step 2, it implies the possibility of overlaps between main branches of sub-trees.

This shifting can be applied to the grouped layout and the bundled layout alike and significantly disambiguates overlapping structures and reduces clutter, as can be seen for example in Figure 3.8 on page 29.

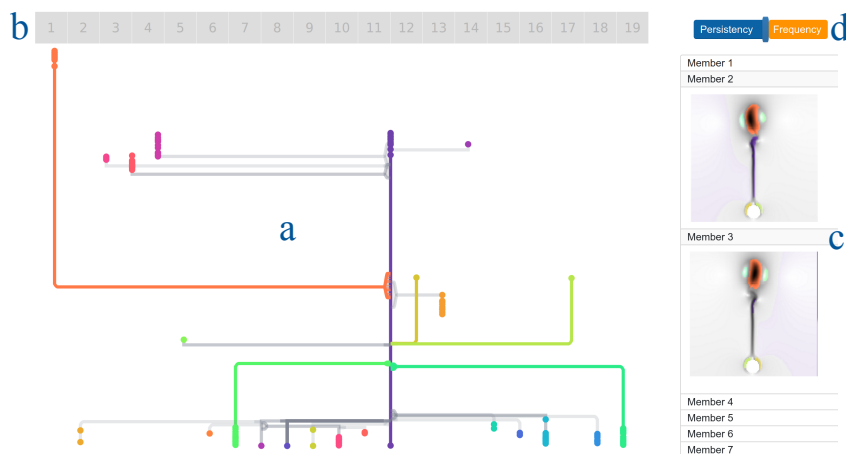


Fig. 3.10.: The Fuzzy Contour Tree user interface: **a** the Fuzzy Contour Tree combines all member contour trees in one visualization. **b** The member grid reflects all members via their number. **c** The component viewer highlights selected components in the fuzzy contour tree in the individual member fields, and **d** the weights slider allows to choose the weights for persistence and frequency that are used during the layout procedure.

3.3.3 Interaction

I implemented the interactive visualization (including the layout algorithm discussed above) in a lightweight JavaScript prototype based on the d3.js library [24]. An overview of the resulting user interface is given in Figure 3.10 on the preceding page. It allows fully fluid interaction for all data sets we consider in Section 3.4. The Fuzzy Contour Tree (a) is the main part of the visualization. The ensemble members whose contour trees are shown in the Fuzzy Contour Tree are represented by numbered boxes on top in the member grid (b). This grid provides the link from the Fuzzy Contour Trees to individual members. The component viewer (c) provides the link between arcs in the Fuzzy Contour Tree and corresponding regions in the member fields. Using the weights slider (d), users can adapt the weights of persistence and frequency that are used in the layout procedure.

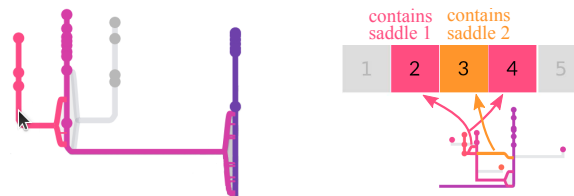


Fig. 3.11.: **Branch highlighting** (left): the selected branch and all ancestors are highlighted with all connected saddles. **Member highlighting** (right): each member containing an edge in the selected bundle is highlighted in the associated color.

We developed **branch highlighting** (left in Figure 3.11): hovering a branch in the Fuzzy Contour Tree (a in Figure 3.10 on the preceding page) highlights this branch and all its bundled edges and ancestors while all other branches are grayed out. The selection can be fixed and released by clicking. For a fixed selection, the ancestors of selected branches are not highlighted to clarify which branches are selected (see Figure 3.12b on the following page).

At the top of the UI, the member grid (b in Figure 3.10 on the preceding page) provides information on individual contour trees. Figure 3.12a on the following page shows **tree highlighting**: selecting the index of an individual contour tree in the member grid highlights this particular contour tree. To clarify membership of each branch, highlighting a branch in the Fuzzy Contour Tree also triggers **member highlighting** in the member grid (on the right in Figure 3.11). All members that contain one of the highlighted edges are colored correspondingly in the member grid. The right part of Figure 3.11 shows member highlighting for a branch with two associated saddles. In this case, each saddle is assigned a different color to clarify the structure of the member contour trees. The highlighted branch is colored in pink, and so is the saddle it is most frequently connected to in the member contour trees.

This saddle is present in members 2 and 4. The second saddle is colored orange and present only in member 3.

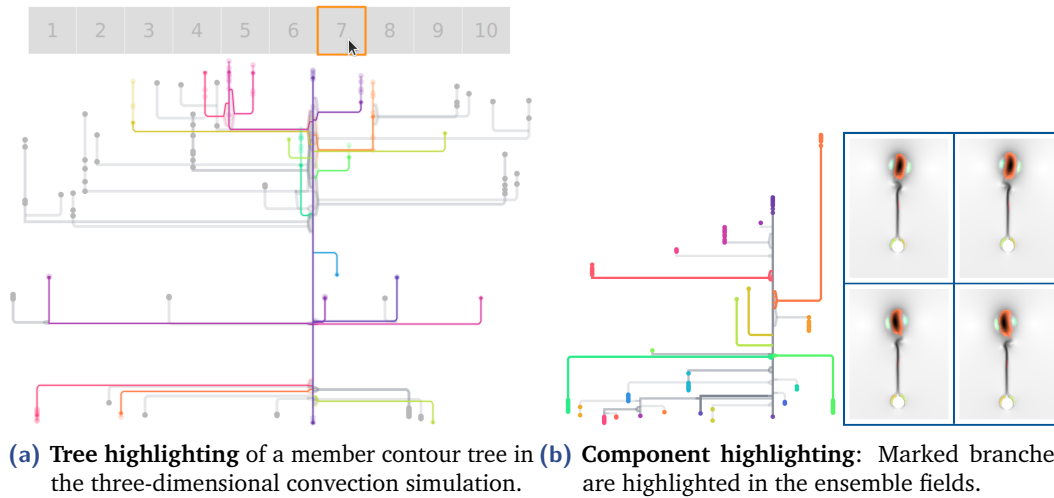


Fig. 3.12.: Tree and component highlighting in the convection simulation in bundled layout (left: three dimensions, right: two dimensions, optimized branch spacing).

It is furthermore sensible to link the Fuzzy Contour Tree to a spatial representation of each of the individual analyzed scalar fields. Like this, a better understanding of the contour tree components is offered, and users are able to check the quality of the matching; if one component from the Fuzzy Contour Tree represents very different areas in the individual members, the alignment might be of poor quality. This link is implemented in the component viewer (c in Figure 3.10 on page 32) as **component highlighting** for the two-dimensional case, as shown in Figure 3.12b. While there is no similar functionality implemented for three-dimensional data sets, volume rendering or isosurface visualization can be applied in these cases.

The weights slider (d in Figure 3.10 on page 32) reflects the ratio between weights for persistence w_{pers} and frequency w_{freq} summing up to 1.

3.3.4 Layout Parameters

Several parameters influence the layout process. The temperature function of the simulated annealing and its parameters are adopted from the layout algorithm by Heine et al. [80] and are discussed there. Additional parameters in our approach are the branch rating R , used to obtain the alignment branch decomposition by using it to weight crossings during the simulated annealing. As described above, this rating depends on the weights w_{pers} and w_{freq} , weighting the influence of the path persistence $P_{\%}$ and the frequency $F_{\%}$.

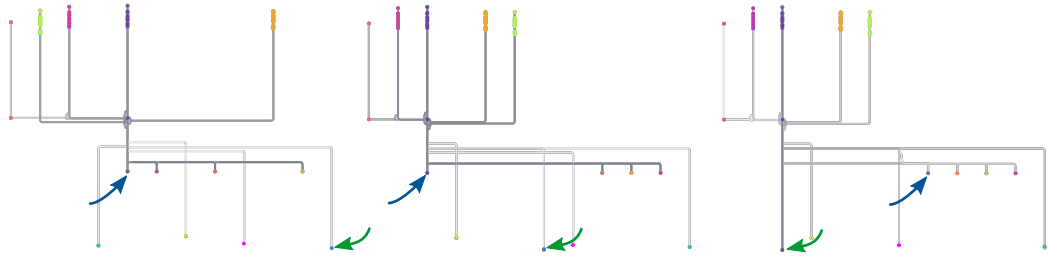


Fig. 3.13.: Different layouts and weights: (from left to right) layout with $w_{\text{freq}} = 1$, another randomization with $w_{\text{freq}} = 1$, the same randomization as in the center, with $w_{\text{pers}} = 1$. In all trees, the same two nodes are highlighted. Ranking branches in the branch decomposition based on frequency, the main branch is contained in all ensemble members and does not span the whole isovalue range of the Fuzzy Contour Tree. The high frequency is visible in the branch opacity. Ranking based on persistence on the other hand, the main branch is the one with highest persistence, spanning the whole isorange, but it is only contained in a single member contour tree (visible via its low opacity in the left two layouts, where it is not the main branch). This makes the layouts resulting from persistence weight difficult to interpret and predict. In the right picture the opacity is chosen due to persistence since it reflects the branch rating.

While it appears natural to consider persistence of branches in the rating, our experiments showed that the most intuitive results are obtained with $w_{\text{pers}} = 0$. Node values in the alignment differ from those in the individual contour trees, hence the persistence in the alignment cannot be considered an intuitive stability measure, making its impact difficult to interpret. An example for this effect is given in Figure 3.13.

3.3.5 Challenges

Several specific challenges arise when visualizing a Fuzzy Contour Tree that are not present in the visualization of individual contour trees.

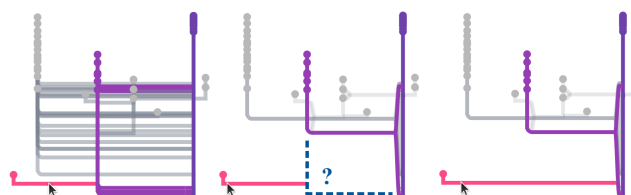


Fig. 3.14.: Challenge in the bundled layout: in the grouped layout (left), the pink branch is connected to its parent. In the bundled layout (middle), this connection becomes lost; the branch needs to be extended (right).

Bundling edges, children that connect to a saddle at an isovalue that is not covered by the bundled branch need to be connected to their parent's bracket. An example

is given in Figure 3.14 on the previous page: in grouped layout, the pink child is connected to its parent branch. Multiple saddles of the parent branch are at higher isovalues than the parent branch this child is connected to. Hence, in bundled layout with the bundled edge at the mean value of all saddle isovalues, the child is no longer connected to its parent (center of Figure 3.14 on the preceding page). On the right, this missing connection was fixed by an extended edge.

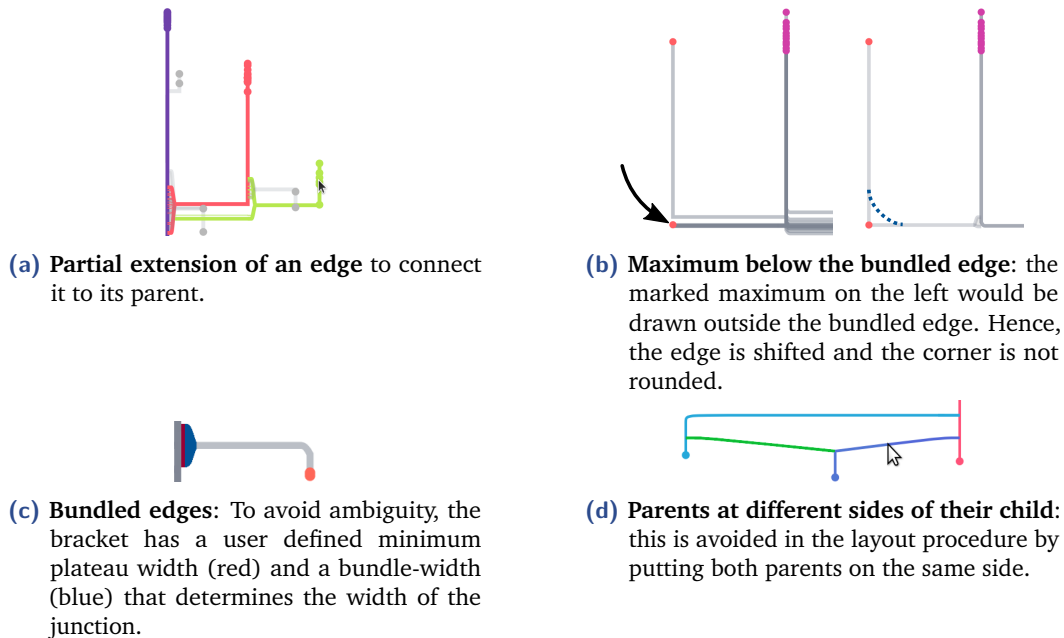


Fig. 3.15.: Different challenges and their solutions in the bundled layout.

A special case of extended edges occurs, when only a part of the child's bracket is not connected to the parent. If this case can not be avoided by shifting the parent's bundled edge, only some of the bundled edges are extended, as shown in Figure 3.15a. In this case, the filling of the bracket is omitted.

In case of children that are connected to the parent's bracket instead of the edge, it needs to be assured that these children are not perceived as being connected to the parent branch of their parent. For example in Figure 3.14 on the preceding page the pink branch needs to be clearly visualized as child of the purple branch, not of the blue branch. This is achieved using the bracket of a minimum thickness. The distance of the bundling point to the origin consists of *plateau width* and a *bundle width*. Both are illustrated in Figure 3.15c and can be customized. In addition, branch highlighting not only highlights the hovered branch, but all its parent branches. This further clarifies the parent-child relation.

In the alignment, it can occur that multiple different saddles appear as the origin of a single branch. In this case, the opacity of branches is determined by the path

occurring in the largest number of individual contour trees, and for each origin minimum and maximum branch are visible. Furthermore, every origin is assigned an individual color for connecting edges. Hence, the existence of multiple origins is emphasized when the affected edge is highlighted, and also in member highlighting, cf. Figure 3.11 on page 33 for an example. When highlighting individual trees, only the edge to the origin occurring in the tree is highlighted. Using the information of individual contour trees, it is ensured in the layout process that all parents are located at the same side of the child branch to avoid far stretched connections to parents like in Figure 3.15d on the preceding page.

To obtain a more pleasing visualization, corners of edges are rounded. In some cases however, this is not feasible: If the isovalues of leaves are too close to the saddle's isovalue, the corner needs to be sharp to ensure that no leaf needs to be drawn on a rounded edge; see the purple branch in Figure 3.15c on the facing page for an example. The same situation occurs if the isovalue of a child's saddle is too close to the isovalue of its parent's saddle.

The metric used in the alignment process ensures that extrema are only matched to extrema of the same type. Nevertheless, the range of aligned extrema can be large, such that individual extrema might lie above or below the mean value of the edge's saddle values, resulting in extrema that would be drawn outside the bundled edge. See Figure 3.15b on the preceding page for an example of this situation. Since the type of matched extrema is unique, the lowest respectively highest matched saddle is always below (for maxima) or above (for minima) the lowest maximum or highest minimum. Hence, the bundled edge can always be shifted towards extrema that would be drawn outside the edge, to avoid this situation.

3.4 Results

Using our approach, visualizing contour trees of scalar field ensembles is straight forward: from each ensemble member, the contour tree is extracted. This allows optional pre-processing of the data to obtain meaningful contour trees such as noise removal or contour tree simplification for each individual member tree – all available research on contour tree creation can be used to improve the Fuzzy Contour Tree. Afterwards, the alignment is computed and the Fuzzy Contour Tree is visualized using the layout strategies and interactions discussed in the previous section are applied.

The application to several analytical and real-world examples in two and three dimensions are illustrated in this section together with the exemplary solution of the visualization tasks to **compare**, **combine** and **separate** ensemble members. Running times for a sequential implementation of the alignment algorithm, formulated as a C++ TTK [194] filter, and general data set properties, are given in Table 3.1. All times were obtained on a standard workstation with an Intel Core i7-7700 and 16GB of RAM. Contour trees were computed and –if sensible– persistence-simplified using TTK.

Data set	Size	n	$ V_{\max} $	$ \mathcal{A} $	t_{align} [s]
outlier/scattered peaks	128×128	16	20	28	0.06
convection 2D	128×256	23	32	62	0.15
convection 3D	$64^2 \times 128$	10	60	144	1.10
viscous fingers	$64^2 \times 45$	15	48	128	0.83

Tab. 3.1.: Properties and runtimes of the example data sets: the ensembles contain n members; $|V_{\max}|$ and $|\mathcal{A}|$ denote maximal contour tree size (after simplification) and alignment size. t_{align} denotes the alignment computation time in seconds.

3.4.1 Analytical Ensemble with Outlier

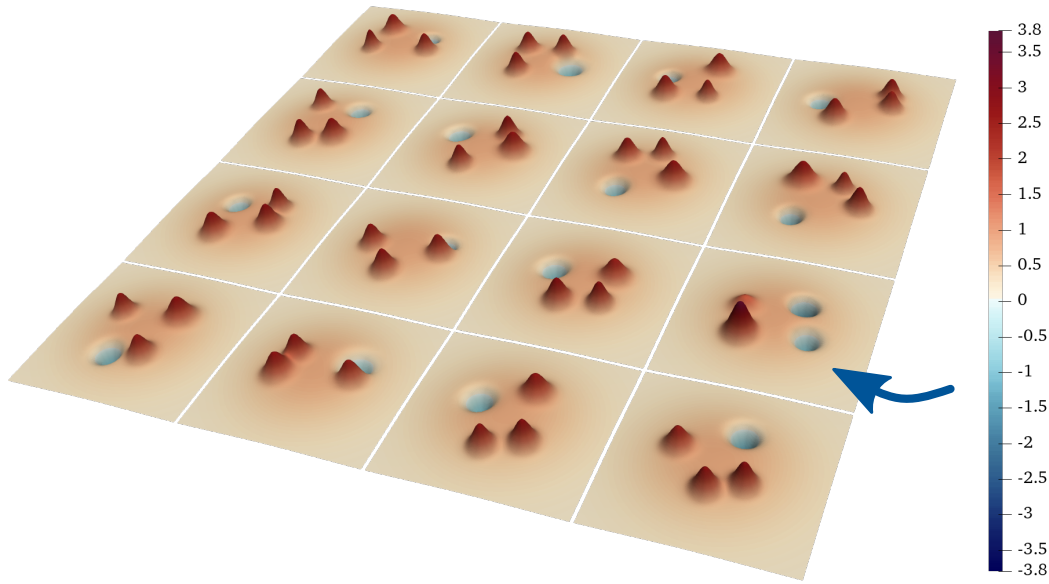


Fig. 3.16.: Ensemble members of the **outlier ensemble**: the highlighted outlier has two minima and maxima instead of three maxima and one minimum.

To demonstrate the usefulness of Fuzzy Contour Trees and give a straightforward example that illustrates both alignment and layout, we created an analytical two-

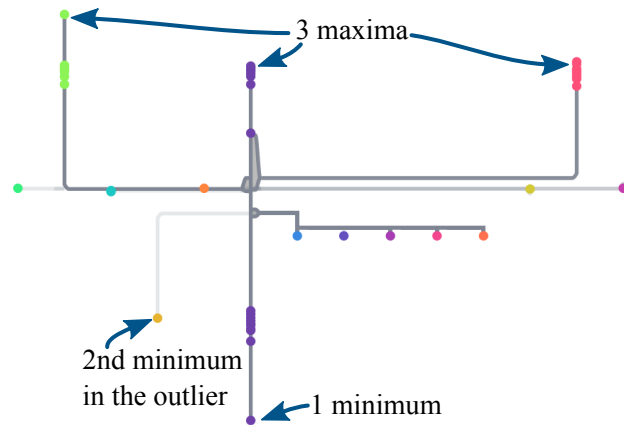


Fig. 3.17.: Fuzzy Contour Tree of the outlier ensemble: structures with high frequency are clearly visible as well as the structure of the outlier.

dimensional data set with simple structure shown in Figure 3.16 on the preceding page: Each of the 16 ensemble members contains a small local maximum in the center and 4 local extrema of varying height around the center peak. In 15 members these extrema are three maxima and one minimum, in one further (outlier) member there are two minima and two maxima. Figure 3.17 shows the Fuzzy Contour Tree for this data set, computed using the persistence metric from Section 3.2 on page 20.

Comparing the different ensemble members using the Fuzzy Contour Tree is straightforward: using the bundled layout, the branches with high frequency are easily determined by their high opacity. Also, the existence of three maxima and one minimum in most ensemble members is clearly apparent, as are the isovalues inducing topological changes.

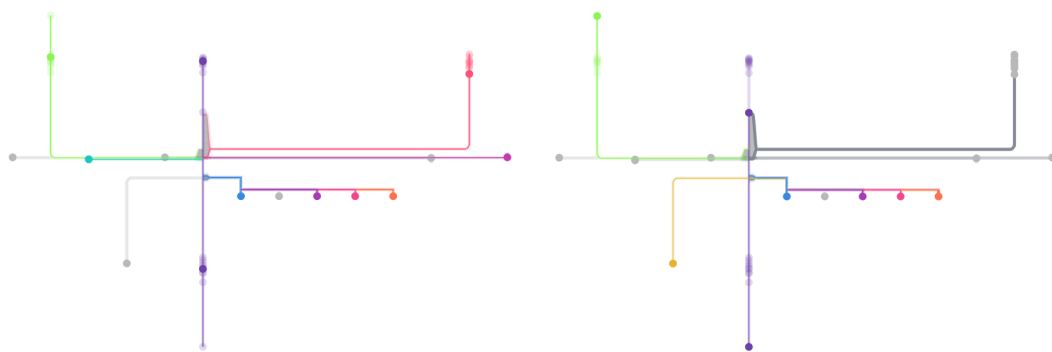


Fig. 3.18.: The differing structure of the outlier in the outlier ensemble is emphasized using tree highlighting: (Left) A typical member is highlighted. Its branches follow branches with maximal frequency. (Right) The dissimilarity of the structure of the outlier is clearly visible.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Fig. 3.19.: Member highlighting in the outlier ensemble of a branch with low frequency (top) and a branch with high frequency (bottom) clearly shows which ensemble member is the outlier. The colors correspond to the chosen minimum and maximum in Figures 3.17 and 3.18 on the preceding page.

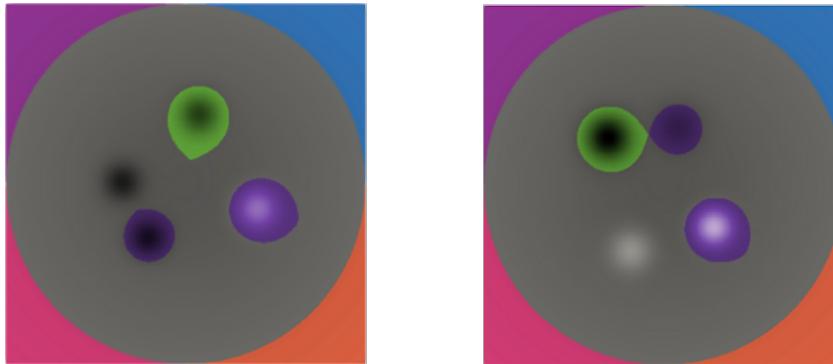


Fig. 3.20.: Component highlighting: selected branches shared by the members in Figure 3.18 on the previous page are highlighted in the ensemble members. Reproduced from [*120].

Combining and identifying the members that share the structure with three maxima and one minimum is possible: Topological structures contained in every member of the ensemble are given by the branches that are contained in the typical member and the outlier in Figure 3.18 on the preceding page; this can be seen by their opacity in Figure 3.17 on the previous page and using member highlighting (Figure 3.19). As expected, two maxima and one minimum are part of the common structure of the whole ensemble, as well as four small, linked branches at the vertical center of the tree, that correspond to the four corners of the domain. This can be seen using component highlighting (Figure 3.20). Note that the small maximum in the center of each ensemble member is not visible as a common structure in this case, but as multiple (nearly) horizontal branches. This results from the super-tree property and a chosen metric for the alignment creation that favors high persistence (cf. Section 3.2.1 on page 22 and Figure 3.5 on page 21).

Separation of the outlier can be accomplished using member highlighting on the single minimum with low frequency on the left of the tree (Figure 3.19). The structure of the outlier in comparison to the other ensemble members is even more apparent when using tree highlighting on the corresponding trees (Figure 3.18 on the previous page). To further investigate the branches that the outlier shares with the rest of the ensemble, these edges can be investigated using component

highlighting: In Figure 3.20 on the facing page, the components corresponding to the shared branches (including the main branch) are highlighted for the outlier and the individual contour tree shown in Figure 3.18 on page 39. Both scalar fields behave similarly in the domain corners. In addition, the maximum of the outlier is matched to different local maxima in the other members, thus explaining the high variance in leaf isovalues in the main branch.

3.4.2 Scattered Peaks Ensemble

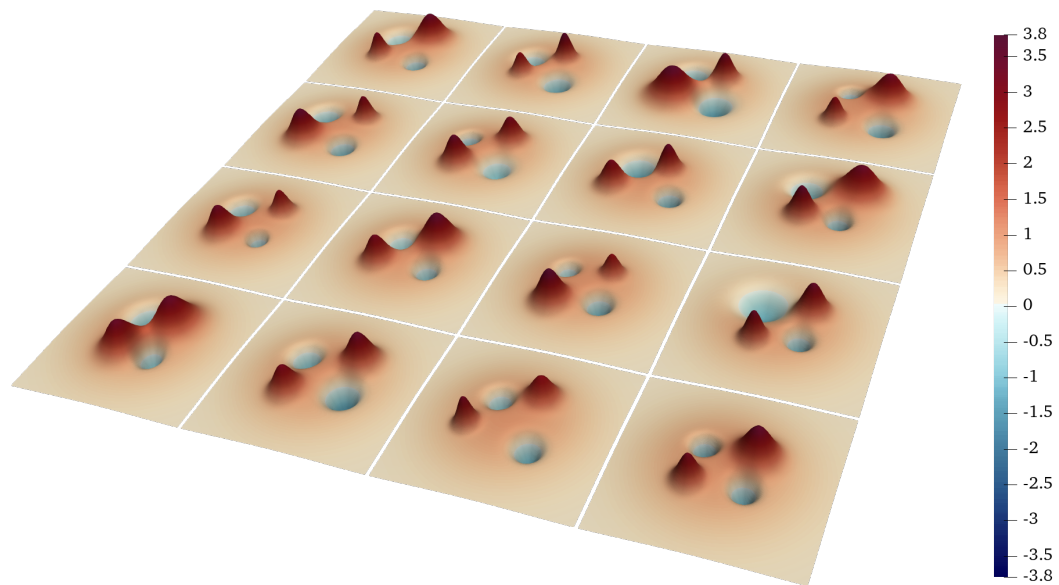


Fig. 3.21.: Ensemble members of the **scattered peaks ensemble**.

The scattered peaks ensemble shown in Figure 3.21 is of a similar structure as the analytical example: all 16 members contain two maxima and two minima in addition to a small peak at the center. The Fuzzy Contour Tree in Figure 3.22 on the next page illustrates the behavior and limitations of the alignment.

As described above, other approaches typically use some sort of overlap measure to map features defined by the contour tree segmentation onto each other. In contrast, our method is independent of the position and area of matched arcs and nodes (unless they are explicitly incorporated in the used metric). If multiple fields share the same main features in a similar topological structure (i.e. their positions and connections in the contour tree), but they are scattered differently over the domain, our method can still find and match them. Naturally, this is only possible as long as the overall topological structure provides a sufficient amount of similarity.

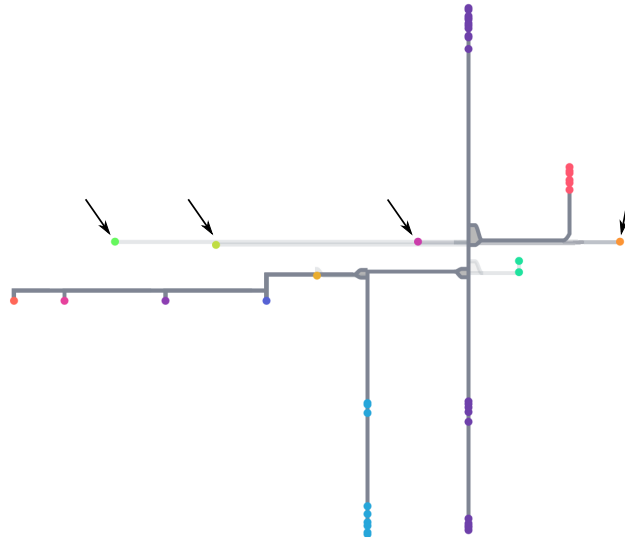


Fig. 3.22.: The **Fuzzy Contour Tree of the scattered peaks ensemble**: the small peak in the center of the domain is present in multiple, unmatched branches to allow matching of the four main peaks.

In this example, the four extrema around the center are identified as the main structure of the ensemble. Due to topological variance the small peaks in the center can not be matched as long as the four main structures are matched. The alignment thus contains multiple different branches representing the small peak. This can be seen in Figure 3.22: the small peak occurs multiple times with a low frequency. This allows the mapping of the two main maxima and the two main minima, even though they are located in completely different areas in their corresponding fields.

The matched minima show some limits of our matching method: as explained in Section 3.2.1 on page 22, there are limits to the matching possibilities based on where certain features are positioned in the tree. If the intuitive match is not possible, the alignment either splits the two features, resulting in two copies of the branch in the resulting alignment, or it matches them to a completely different part of the tree, with the latter typically being the less intuitive or desired option.

Both cases can be observed in Figure 3.22: big and small maxima in the ensemble are matched according to their size, whereas the two sizes of minima are not separated. The small peak at the center can not be matched over the ensemble members either. Here, the feature is split into several copies in the Fuzzy Contour Tree, each having a low frequency.

In the Fuzzy Contour Tree, it is easy to find the four extrema that are matched by all members in the alignment and the four corners of the domain when choosing the branch opacity according to the number of occurrences of branches in member

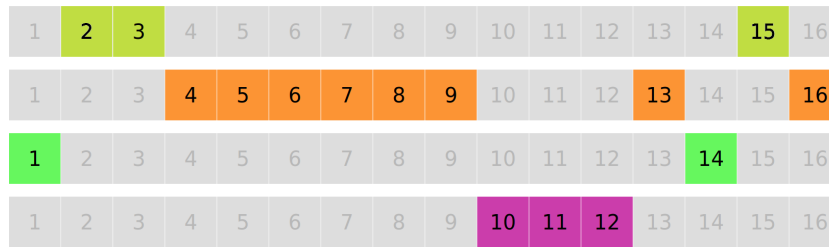


Fig. 3.23.: Member highlighting shows the presence of one small peak in every member.

contour trees (Figure 3.22 on the preceding page). The lower opacity and identical height of the branches representing the small peak indicate the “splitting” of this peak for different members. Using member highlighting, it is easy to see that such a small peak is present in every ensemble member (Figure 3.23).

3.4.3 Convection Simulation

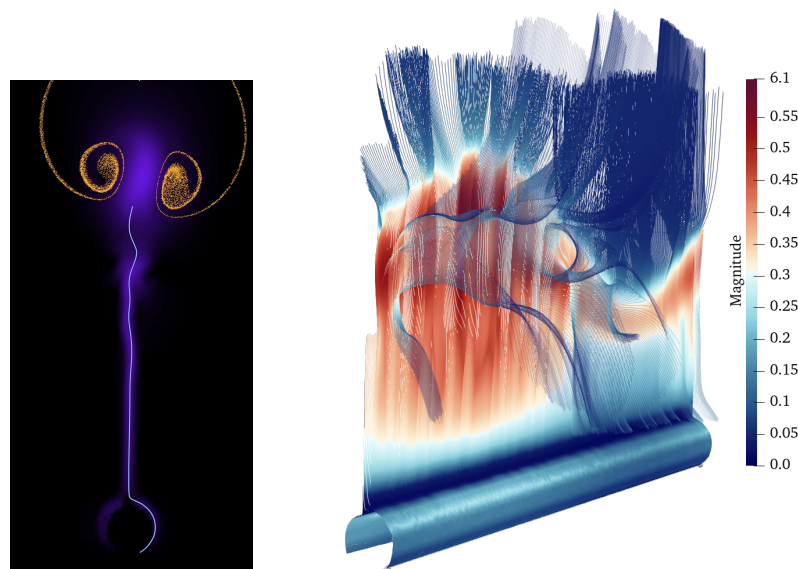


Fig. 3.24.: Exemplary members of the **convection simulation** in two (screenshot from [121]) and three dimensions.

The convection simulation ensemble describes the flow around a heated pole in two and three-dimensional domains; exemplary members are shown in Figure 3.24. The ensemble was obtained by simulating the corresponding model with stochastically perturbed initial and boundary conditions for velocity. Fluid that is initially at rest is heated around the pole, begins to rise, and forms a plume. Scalar values describe flow vorticity, and topological segmentation identifies vortices as the attracting basins

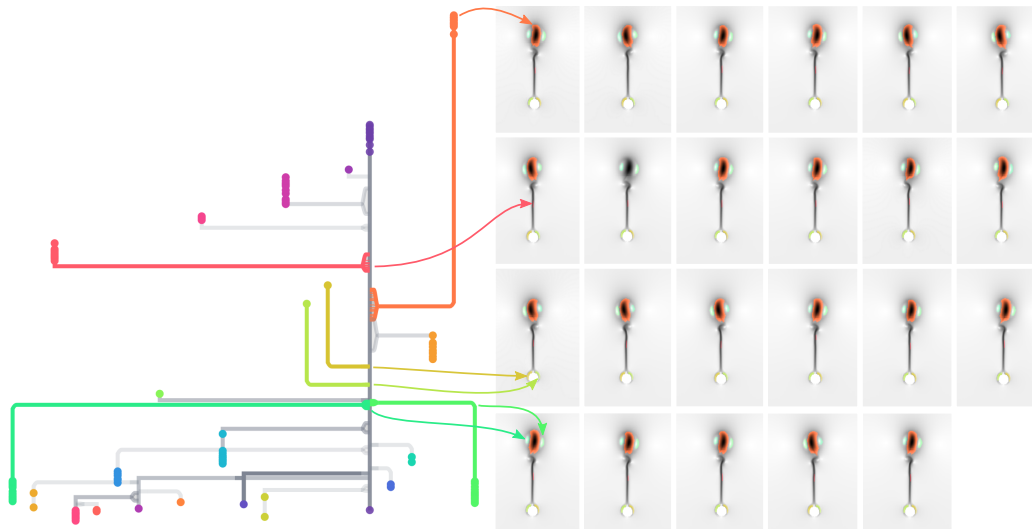


Fig. 3.25.: Component highlighting shows the high quality of the matching of components in the alignment of the two-dimensional convection simulation. In the Fuzzy Contour Tree, branch spacing is optimized.

of maxima. The contour trees for both data sets were simplified using persistence, with the same threshold for all members.

Two-Dimensional Ensemble. Using the combined cost metric (cf. Section 3.2 on page 20) for the alignment of the ensemble contour trees results in a highly intuitive matching, as can be verified in the component view: several highlighted components across the ensemble members are shown in Figure 3.25. For example, the global maxima at the center are matched onto each other over (almost) all members. They are represented by the marked orange branch in the Fuzzy Contour Tree with small variance in the matched saddles and leaves. Based on the intuitive matching, it is easy to **compare** all ensemble members and identify common topological structures.

Looking at the Fuzzy Contour Tree without optimized branch spacing in Figure 3.26 on the facing page, above the branching of the orange global maxima, several smaller peaks occur on the left hand side of the Fuzzy Contour Tree (see Figure 3.26 on the next page). In each member contour tree, zero to three of them occur. This can be verified using tree highlighting, allowing to **combine** or **separate** members based on the occurrences. Other examples for intuitively matched features are the (green) minima left and right of the main maximum and the two maxima surrounding the cylinder on the bottom. All of them are highlighted in the Figure.

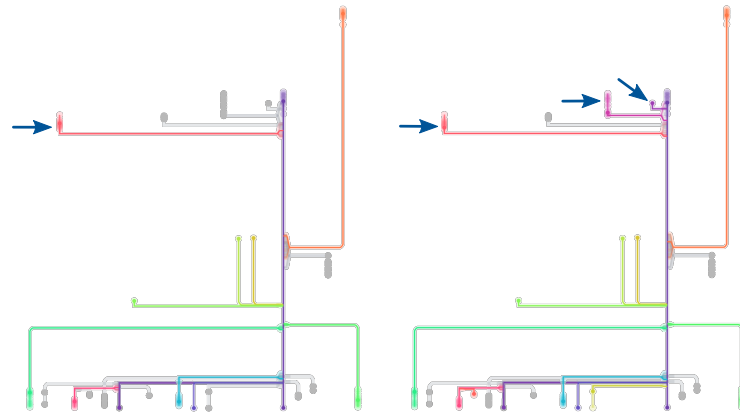


Fig. 3.26.: Tree highlighting: (left) tree number 3 only contains one maximum on top of the Fuzzy Contour Tree while tree 8 contains three of them.

Three-Dimensional Ensemble. Also in the three-dimensional case, using the combined cost metric for the alignment provides intuitive results (Figure 3.27 on the following page). **Comparison** of the topological structure hence provides a clear insight in common topological structures of the ensemble members: at the bottom, only minima exist, then a layer of maxima occurs, followed by another area with mainly maxima; these extrema indicate vortices of different rotational direction. Whether this structure is present in all members can be checked using tree highlighting, providing a common segmentation for the relevant members to **combine** them. The single blue minimum (vortex) between the two layers of maxima **separates** one ensemble member from all others. The number of this member can be determined easily using member highlighting for the branch (Figure 3.28 on the next page).

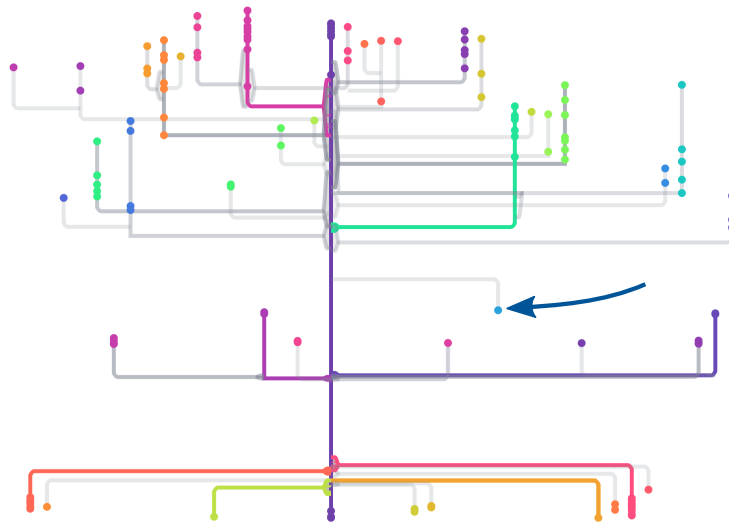


Fig. 3.27.: Common topological structures of the three-dimensional convection simulation ensemble are clearly shown in the Fuzzy Contour Tree. All branches occurring in at least 8 out of 10 members are highlighted.

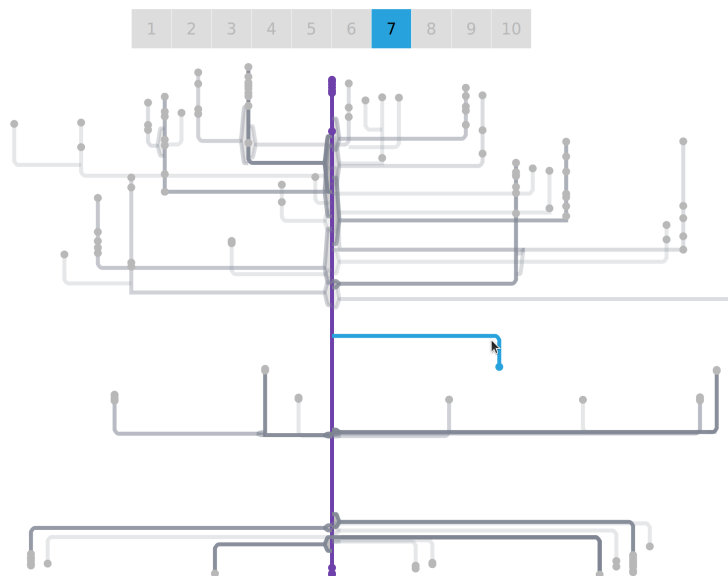


Fig. 3.28.: Separation of members that contain singular structures is possible by member highlighting in the Fuzzy Contour Tree of the three-dimensional convection simulation.

3.4.4 Viscous Fingering

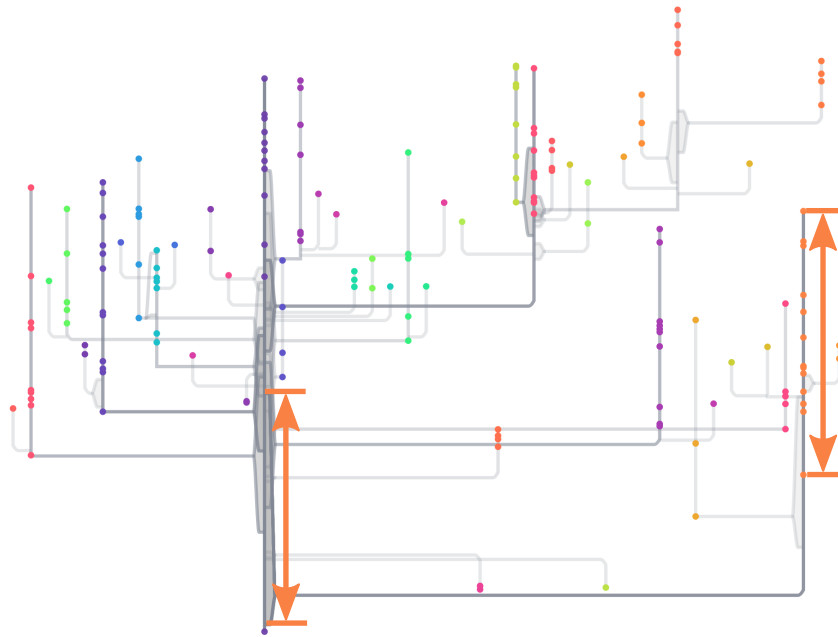


Fig. 3.29.: Large differences between saddles and leaves in a single branch indicate the matching of semantically unrelated branches for the viscous fingers ensemble. Ranges for the orange branch are marked. Despite the low topological similarity and the resulting high number of branches in the Fuzzy Contour Tree, bundling and optimized branch spacing allows a comprehensible visualization.

To illustrate the behavior of our method in a setting where searching for topological similarities in the member's level sets is not meaningful, we consider the Fuzzy Contour Tree for 15 members of the viscous fingering ensemble [172].

The viscous fingering ensemble was provided for the 2016 Scientific Visualization Contest and contains 50 simulations of a viscous fingering process, where scalar values represent salt concentration and topological segmentation identifies individual fingers [123]. From the time-varying ensemble, three members and five consecutive time steps are chosen, resulting in an ensemble of 15 contour trees. We follow the approach of Lukasczyk et al. [123] to derive three-dimensional piecewise linear scalar fields from the given point clouds. Because it is not clear when and where viscous fingers appear and how they evolve, the variance between ensemble members is very high in this data set. Hence, searching for topological similarities is not meaningful.

As Figure 3.29 illustrates, the Fuzzy Contour Tree for the viscous fingering ensemble is highly complex. The large variance in the scalar values of matched critical points and the distinct overlap of the value ranges indicates that the matching is not

semantically meaningful. While this example is beyond the limits of the Fuzzy Contour Trees, it shows that an identification of non-meaningful alignments is possible using only the Fuzzy Contour Tree.

3.5 Discussion

As shown in the previous section, Fuzzy Contour Trees are useful to visualize topological structures across ensembles. Fundamentally, tree alignment, i.e. the matching of individual contour tree nodes and arcs into a super-tree enables the joint layout of all contour trees as a Fuzzy Contour Tree. However, it also imposes some limits w.r.t. possible applications: often, overlap measures are used to map features defined by the contour tree segmentation onto each other. In contrast, our method can be independent of position and area. If the same major features are shared among multiple members in a similar topological structure (regarding relative positioning and connectivity in the contour tree), our approach is able to find and match them, even if they are scattered differently over the domain without overlap. Naturally, this is only possible as long as the overall topological structure provides a sufficient amount of similarity for a meaningful matching.

If the structure of the different contour trees shows only small or no topological similarity –as discussed in the viscous fingers example above– a minimal alignment will exist (and is computed by our algorithm), but the matching will not be meaningful, resulting in a non-meaningful visualization.

Up to now, the automatic identification of semantically meaningless alignments is not possible with our method. While it would appear intuitive to consider the alignment cost as a criterion and declare the alignment as unsuccessful if the cost is too high, this cost is a heuristic that does not allow an absolute comparison. Especially it can not be generalized across different data sets. However, identifying alignments containing matchings of unrelated topological components can be achieved by a user when comparing matched segments via component highlighting, and by finding indications in the Fuzzy Contour Tree such as large differences in vertical coordinates of saddles and leaves with large overlap.

Comparison to Similar Techniques. Compared to the combined visualization of the Fuzzy Contour Tree, displaying multiple contour or merge trees side-by-side provides much less support for the basic visualization tasks to **combine**, **compare** and **separate** members. Independent visualization of individual contour trees results

in different layout and scaling; thus, a sensible comparison of the contour trees, especially of value ranges, is not feasible. Even if identical layout and scaling could be obtained, there are strong limits on the visual scalability of a side-by-side approach, and manual or “visual” matching of subtrees has to be performed by a viewer, making the approach non-practical overall.

Favelier et al. [56] cluster ensemble members based on an embedding of their persistence maps in Euclidean space. Using the notion of mandatory critical points, confidence regions for each cluster are calculated and visualized. Athawale et al. process a given two-dimensional Morse complex ensemble to obtain a probabilistic map and a survival map, called *summary maps* for two-dimensional Morse complex ensembles [10]. The probabilistic map shows the probabilistic classification of all points in the plane based on the mandatory maximum their integral curve ascends to over all ensemble members, while the survival map traces the behavior of gradient flows under persistence simplification, where unchanged gradient flow direction after a simplification step is counted as a survived step.

Both techniques are suited for ensembles of arbitrary size, but do not consider or present single or combined contour trees. Furthermore, a Fuzzy Contour Tree incorporates information from individual contour trees into a single overall visualization, and links this combined visualization back to the individual contour trees; this possibility fundamentally enables **separation** and is not available in either summary maps or the persistence atlas. While the persistence atlas provides **combination** of the ensemble members, users are not provided sufficient information on individual members to identify those with common segmentations, unless they are part of the ensemble’s common topological denominator.

A further limitation of the two approaches is the fixed comparison metric. While the persistence atlas relies on trend and location of critical points, and the approach by Athawale et al. is based on the gradient field, our approach *can* incorporate these parameters when matching nodes in the alignment, but it also can be based on the topological structure or any other parameters. This flexibility makes Fuzzy Contour Trees highly adaptable to domain-specific needs.

3.6 Conclusion

By combining tree alignments with a novel layout algorithm, we are able to combine multiple contour trees of ensemble members in one Fuzzy Contour Tree. The resulting visualization is semantically meaningful with minimal clutter. Together with

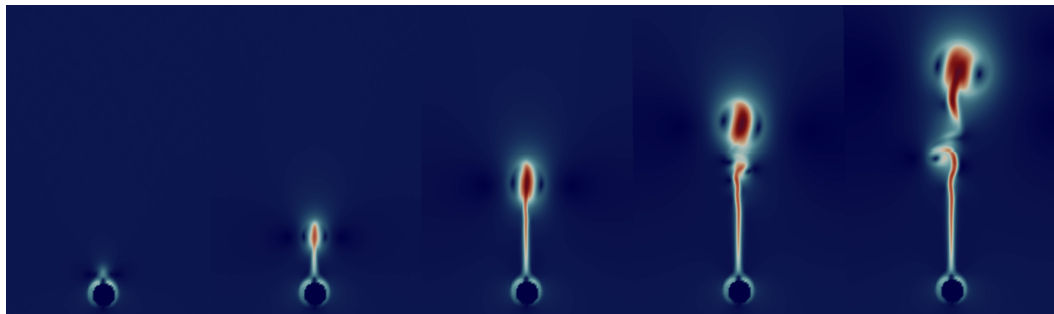
its interaction possibilities, it allows the **comparison**, **combination** and **separation** of ensemble members based on topological features. For future research, there are several opportunities:

While the current algorithm works well in practice, the *deterministic computation* of minimal tree alignments would enhance the stability of our approach. Also, the *automated identification* of non-meaningful alignments would improve the confidence in our technique in real-world use.

To further reduce clutter, different options could be considered: hiding nondescript branches, additional abstraction, and detail on demand to name but a few. In addition, instead of a purely additive comparison of contour trees, a subtractive comparison just showing branches that are not contained in a set of members could improve the overview.

Combining our approach with image databases [2] for in situ visualization is an interesting opportunity. In addition, our progress in simultaneous layout of multiple contour trees can be beneficial in other scenarios involving several contour trees; for example regarding time-dependent scalar fields. This scenario is considered further in the following Chapter 4.

Uncertain Topological Features in Time-Dependent Scalar Fields



Simulations and recordings of physical events often result in series of records over time. Analyzing this data is usually done feature based, by identifying regions with specific patterns, which are then tracked between time steps. While these approaches are helpful for specialized tasks, they are not suited to provide a general overview of the complete data set due to their focus on specific features and the chronology.

Viewing time-dependent scalar fields as ensembles that depend on the parameter “time”, their visualization using Fuzzy Contour Trees is straight forward. The result provides a holistic view of the data set, giving a simultaneous overview of the topological structure of the time-dependent data set.

Similar to the original Fuzzy Contour Tree, the *time-varying Fuzzy Contour Tree* gives insight in the topological structure of multiple fields by simultaneous visualization of the corresponding contour trees. In cooperation with Frederike Gartzky, Florian Wetzels, Luisa Vollmer and Prof.Dr. Christoph Garth, I enhanced the Fuzzy Contour Tree back- and front-end to adapt it to the specific needs of time-dependent data analysis. Together with Frederike Gartzky and Luisa Vollmer, I treated the enhanced Fuzzy Contour Tree layout and the improved member grid in their masters project. Adaptions in the backend –the alignment procedure– were made by Florian Wetzels and Christoph Garth. Time-varying Fuzzy Contour Trees were published at IEEE Vis 2021 [*118].

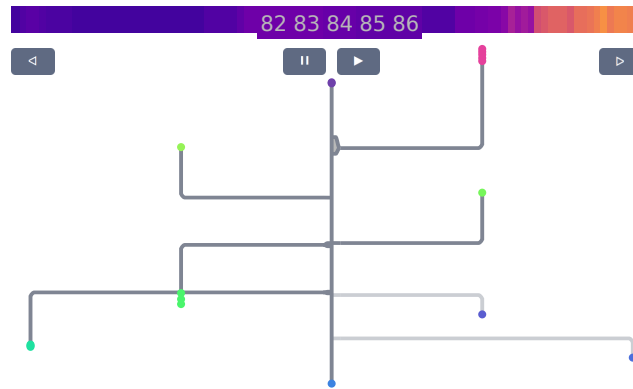


Fig. 4.1.: The time-varying Fuzzy Contour Tree interface: Time selector on top and the Fuzzy Contour Tree of the selected time steps.

In addition to the tasks that can be solved using Fuzzy Contour Trees, time-varying Fuzzy Contour Trees allow the solution of tasks that are specific to the analysis of time-dependent data. It is for example possible to

- compare** ensemble members to identify which topological features appear periodically,
- combine** time steps with similar contours and track down where they change, and to
- separate** time steps that contain a specific branch from others.

By defining *sub-alignments* that originate from the *overall alignment* —that is the alignment containing all time steps– the analysis of arbitrary sub sets is possible. Hence, the time-dependent data set can be analyzed, and the basic visualization tasks can be solved independent of the chronology.

Figure 4.1 shows the time-varying Fuzzy Contour Tree interface. Both, the time selector on top and the Fuzzy Contour Tree below are enhanced versions of interface components of the original Fuzzy Contour Tree interface, providing specialized interaction with time-dependent data and ensuring a consistent visualization.

In the following, I describe the theoretical background of this visualization and how to generate it. Changes in the back-end of Fuzzy Contour Trees are described in Section 4.2, enhancements of the front-end in Sections 4.3 and 4.4. I implemented the time-varying Fuzzy Contour Tree interface using D^3 , python and jupyter notebook.

4.1 Background: Topology Visualization on Time-Dependent Data

Like the generalization of topology-based methods to uncertainty, their generalization to time varying data is a challenging task. An overview was given by Heine et al. [79]. Recently, most contributions visualize the topology of time-dependent two-dimensional vector fields using feature tracking for singularities and closed streamlines [84, 192, 196]. But also for scalar fields there are considerable contributions:

Time-Dependent Contour Trees. A theoretical consideration of time-varying reeb graphs for continuous space-time data was given by Edelsbrunner et al. [54]. Szymczak describes sub-domain aware contour trees and uses them to track accumulated topological changes between slices of the data set. While the evolution of iso-surfaces is plotted, the contour trees are not visualized [189].

Also different interactive tools for the analysis of time varying contour trees and iso-contours have been developed: An interactive exploration tool for split/merge and contour trees for different time steps was developed by Sohn et al. [182]; they define a topology change graph and use it to navigate trees of individual time steps. Bajaj et al. provide multiple calculated signature graphs on time-varying scalar fields. With different interaction possibilities and the additional visualization of single contour trees, they provide real-time exact quantification in the visualization of iso-contours [11]. Kettner et al. take this idea further to non-decomposable topological properties and higher dimensions in the Safari interface [102]. Lukasczyk et al. define and visualize spatio-temporal Reeb graphs to extract and visualize trajectories and relationships of hotspots [124]. Oesterling et al. show the evolution of extrema in high-dimensional data by plotting a one-dimensional landscape profile for each time step and connecting peaks to indicate critical events [148]. These events are determined as structural changes in time-varying merge trees.

In contrast to time-varying Fuzzy Contour Trees, none of these approaches visualize contour trees for more than a single time step or show the evolution of contour trees.

Feature Tracking in Time-Dependent Data. Depending on the application, different features are of interest. Their definition can be application driven, an example are specific fingerprints of global climate patterns [99] or vortex definitions in

flow visualization. Feature definitions can also be application agnostic, like the topological structure that is visualized in time-varying Fuzzy Contour Trees.

Features have been identified and tracked over time in many different ways. The components of interest can be superlevel sets [124, 182] or sub-domains with special geometric and topological properties [25, 111, 169]. Tracking of the features is often achieved using spatial overlap in time [18, 123, 169, 182]. Other approaches base on topology by tracking critical points and using the persistence of topological properties [25, 211]. Multiple features on different levels of interest are often tracked in tracking graphs [124, 125, 182, 211]. These tracking graphs visualize the evolution of features by keeping track of split and join events, as well as birth and death of components.

In all these cases, determined features are tracked between subsequent time steps, tracing these features over time. Our approach on the other hand does not aim to track individual features, but provides an overview over topological features of the complete data. Considering the topological structure of the complete time-dependent data set and matching it, we provide a more holistic view on the data. Similarities and differences of time steps can be determined flexibly and independent of their adjacency.

4.2 Tree Alignment of Time-Varying Contour Trees

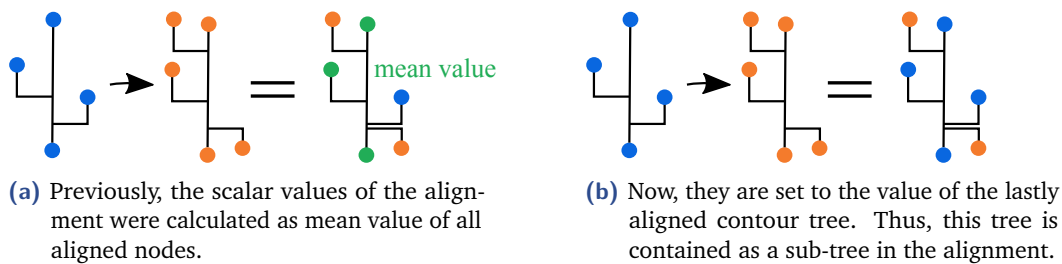


Fig. 4.2.: Different scalar value choices in the alignment.

Considering time-dependent scalar fields, the main focus lies on a consistent alignment over time. Hence, matching of nodes in consecutive time steps needs to be consistent, prohibiting a randomized order of the input contour trees as it was described for the original Fuzzy Contour Trees in Section 3.2.4 on page 25. Instead, the contour trees are aligned sequentially. Furthermore, the matching of nodes in consecutive time steps is prioritized by adapting the choice of values assigned to

the alignment nodes. In the non-time-dependent setting, the values of the alignment are calculated as mean value of the aligned nodes (Section 3.2.3 on page 24). Handling time-dependent fields, the values are set to the value of the lastly aligned contour tree as illustrated in Figure 4.2 on the facing page. Thus, this contour tree is effectively contained in the alignment as a sub-tree with its original scalar values. Aligning consecutive time steps hence takes place in a similar setting to matching adjacent trees and consistent matching of nodes in consecutive time steps is enforced.

The alignment process is very flexible in its application due to the opportunity to use different metrics. For time-varying Fuzzy Contour Trees, we implemented an overlap metric in addition to the existing volume, persistence and combined metrics. The overlap metric is defined as $1 - J(A, B)$ where the Jaccard index of sample sets A and B is defined as

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

The contour tree alignment algorithm with overlap metric and optional matching over time is available in the TTK development branch [194].

Considering the Fuzzy Contour Tree based on the overall alignment provides information on the frequency of topological structures over all time steps. With this information, the tasks to **compare**, **combine** and **separate** records based on their topological structure can be addressed regarding the structure of the complete data set. However, analysis of time-dependent data often takes place on the level of individual time steps instead of the whole data set. Comparing the topological structure of arbitrary sub-sets is rendered possible by the calculation of the corresponding sub-alignment.

Sub-alignments are calculated based on an existing overall alignment. To obtain a sub-alignment, contour trees of time steps that are not selected are one by one subtracted from the overall alignment by decrementing the frequency of all contained nodes. For all nodes that are assigned frequency 0 in this process, all edges and neighboring nodes are considered and connectivity of the result is restored. For less than three edges that are connected to the deleted node, this is straight forward. To connect the neighboring nodes in the case of three and more edges, we determine parent - child relations of the neighboring nodes by determining paths from these neighbors to the fixed alignment root (described in Section 3.2.2 on page 23) and connect the nodes accordingly. The overall alignment as mutual base for all sub-alignments ensures the consistent matching of nodes in different sub-alignments, avoids problems with node identification between sub-alignments and guarantees a consistent layout.

The resulting sub-alignments are alignments of the chosen sub-set of contour trees. They are likely to perform worse with respect to the chosen metric than the result of the alignment algorithm heuristic. However, all we need sub-alignments to be is a sub-tree of the overall alignment to which we can transfer the layout. Edges in the alignment are treated as paths in the Fuzzy Contour Tree layout and actual edges are always verified in the individual contour trees.

Employing the resulting time-varying Fuzzy Contour Tree, arbitrarily selected subsets of the time steps can be **compared**, **combined** or **separated**.

4.3 Layout of Time-Varying Fuzzy Contour Trees

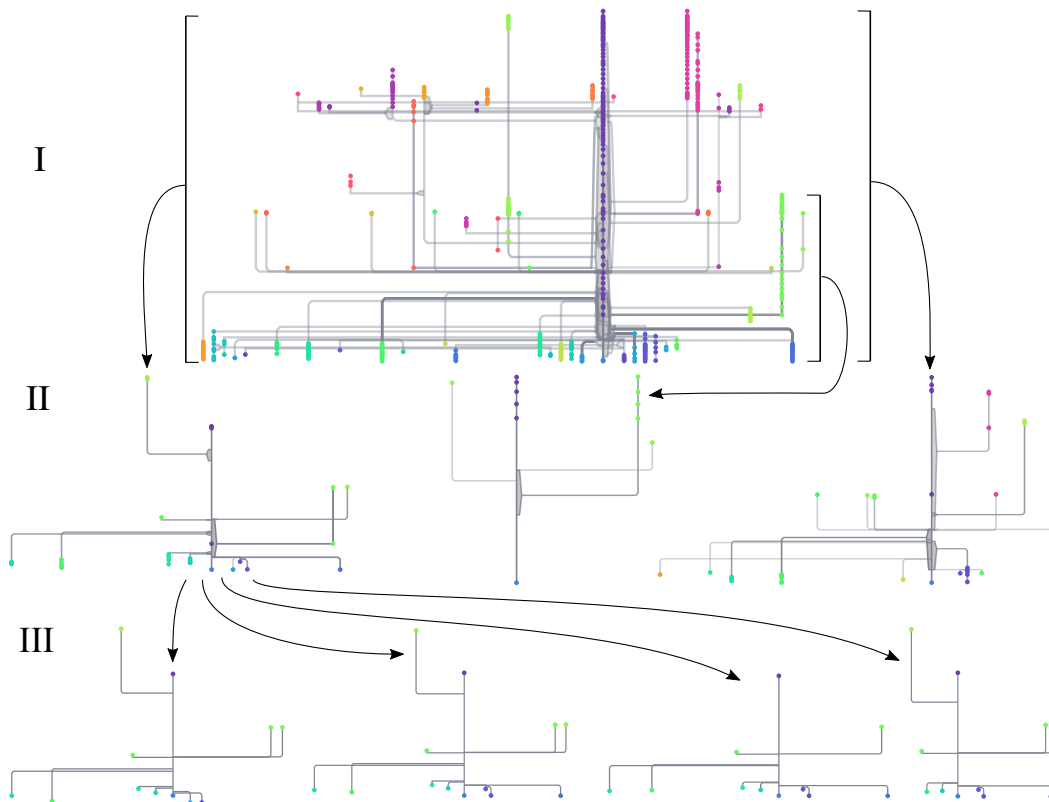


Fig. 4.3.: The trickle-down-layout of the time-varying Fuzzy Contour Tree: the layout of the overall alignment (**Level I**) is applied to sub-alignments (**Level II**). Their layout is propagated to the individual contour trees (**Level III**). The extent of the sub-alignments in the alignment are marked with brackets. Sub-alignments and trees are exemplary. (Reproduced from [*118])

As a basis for the time-varying Fuzzy Contour Trees, the layout of the overall alignment is computed as described in Section 3.3.2 on page 29. As an alteration

to this approach, the used cost function of the simulated annealing not only takes scalar values of leaves and the persistence of branches into account, but also their existence in time: only contemporary branches with overlapping bounding boxes are treated as overlapping.

Similar to the layout of individual contour trees being based on the layout of their alignment, the sub-alignments' layout is based on the layout of the overall alignment. This is possible since the sub-alignment is always a sub-graph of the overall alignment. An illustration for this “trickle-down layout” is given in Figure 4.3 on the preceding page.

To obtain the layout of a sub-alignment from the overall alignment, first the branch decomposition of the sub-alignment is computed. After that, the order of its branches is set according to the horizontal order in the overall alignment.

The main branch is treated as a special case, since it contains two leaf nodes: the designated root node, which is consistent in all sub-alignments, and a leaf node, which can be different for different sub-alignments. Hence, the position of the main branch could either be linked to the root node or the leaf node. To account for the possible changes in its composition, we link the position of the main branch to the leaf node, meaning the main branch will change position according to the position of the leaf node. This not only facilitates spotting changes in the main branch, but also keeping track of its composition.

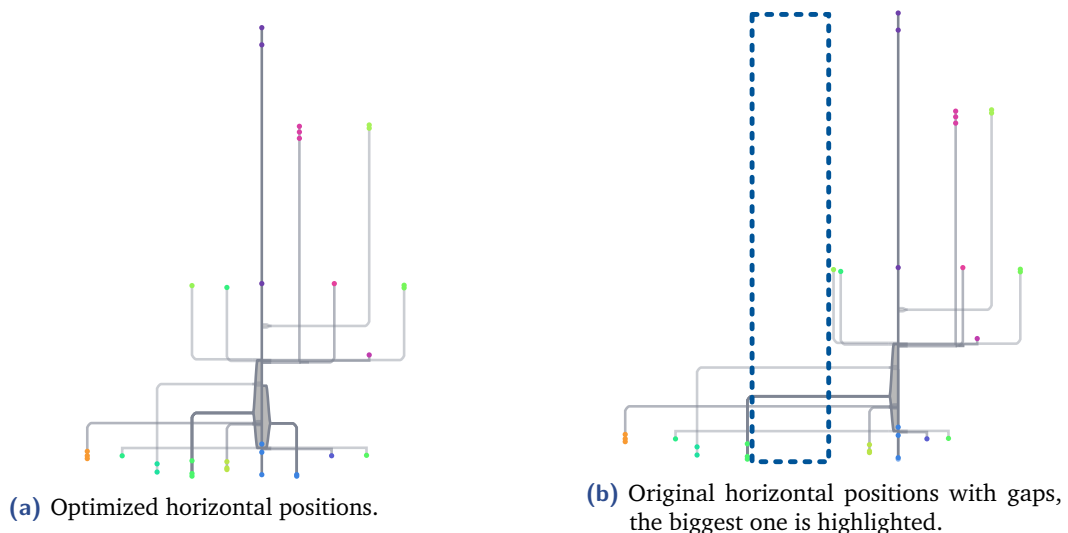


Fig. 4.4.: Options for the **horizontal positioning**.

With the horizontal order of the branches set, there are two options as to how the horizontal position of a branch is determined (Figure 4.4): the first option is to keep

the horizontal position of a branch the same as it is in the overall alignment. This supports keeping track of the individual branches over different sub-alignments, since corresponding branches in the Fuzzy Contour Tree are always drawn at the same position, which is particularly beneficial for animation. However, gaps can occur if branches from the overall alignment are not present in the sub-alignment, resulting in empty positions in the layout. The other option is to allow the horizontal position of a branch to change to have a more uniform layout of the individual Fuzzy Contour Tree while keeping the horizontal ordering of branches. In this case, the distance between two neighboring branches in the Fuzzy Contour Tree of every sub-alignment is constant. For each branch it is checked, if its vertical extent overlaps any of the vertical extents of its direct left neighbors. If there is no overlap, the branch is shifted left to the same position as these neighbors.

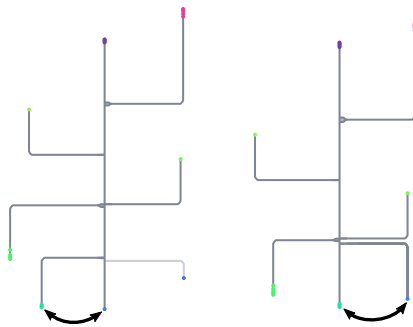


Fig. 4.5.: Two-colored main branch: To highlight changes in the main branch and support tracking the development of branches, the extrema in the main branch are colored differently and consistently.

After the layout for the sub-alignment is determined, leaves, saddles and branches are transferred from the individual contour trees following paths in the sub-alignment, as described for Fuzzy Contour Trees in Section 3.3.2 on page 29. In contrast to the approach described there, coloring is now based on the overall alignment, not on the alignment that is presented in the Fuzzy Contour Tree: we keep the coloring of the leaf nodes consistent by assigning colors to every node id, independent of the node's occurrence in the current sub-alignment. As a further change in coloring, this is also done for both leaves contained in the main branch independently, resulting in two different leaf colors. The coloring stays consistent even if the main branch changes between Fuzzy Contour Trees for different sub-alignments, making it possible to trace the membership of different leaves in the main branch (see Figure 4.5).

Grouped and bundled layout with and without optimized branch spacing carry their benefits over to the time-varying application of Fuzzy Contour Trees and are available as described in Section 3.3.2 on page 29.

4.4 Interaction

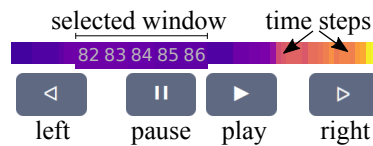


Fig. 4.6.: The time selector: The complete data set is shown with the selected time steps highlighted. Coloring of the time steps is determined by different measures on the sub-alignments to guide the data analyst to patterns or regions of interest. Left, right, play and pause buttons allow the manipulation of the current selection.

Interaction possibilities with the Fuzzy Contour Tree aim on linking information concerning individual contour trees with information on the overall behavior of the considered ensemble (Section 3.3.3 on page 33). While branch and component highlighting remain untouched from the specialization to time-dependent fields, the member grid –enabling tree and member highlighting– was completely revised and turned into the time selector (Figure 4.6). Focusing on enhancements of interactions, component highlighting is not implemented in our prototype.

4.4.1 The Time Selector

The time selector is the enhanced version of the member grid for Fuzzy Contour Trees, providing information on individual contour trees (compare **b** in Figure 3.10 on page 32). Instead of showing only time steps that are contained in the current Fuzzy Contour Tree, every time step from the data set is represented by a colored slice of the selector. Selected time steps are highlighted as boxes with the number of the time step.

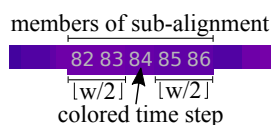


Fig. 4.7.: Coloring based on sub-alignments: the color of time point 84 is determined based on the sub-alignment of the window with 84 at the center. The window size w is user defined.

Coloring Measures. To indicate areas of interest in the time selector, we colored its time steps based on the sub-alignment containing the considered time step as the center of a time window of given width (Figure 4.7). There are two options for this coloring based on centrality measures that are calculated per node of a

sub-alignment:

Showing measure values, the averaged sum of the chosen measure over all nodes of the considered sub-alignment determines the color. Hence, the coloring of a time step gives only information about this single time step and its surrounding window. Showing sub-alignment distances, besides the sub-alignment of the window centered at the current time step, the sub-alignment of the window centered at the subsequent time step is considered. The chosen measure is determined for all nodes of both sub-alignments. The color is determined via the averaged sum of the node-wise difference, where the nodes are paired by the overall alignment. This coloring of a time step provides information about the similarity or dissimilarity of the considered sub-alignment and the subsequent one.

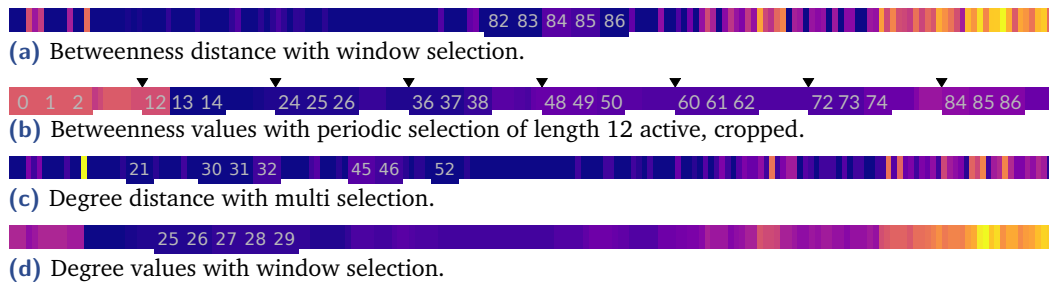


Fig. 4.8.: Time selector options: Selection modes and different coloring measures on the convection simulation data set.

Showing the measure values is helpful to discover structures in the data, while showing differences emphasizes changes between consecutive sub-alignments, highlighting time steps with potentially crucial changes. As possible measures, we implemented two centrality measures: the degree centrality and the betweenness centrality. In contrast to the graph edit distance, the centrality distance is based on the node centrality and therefore takes a weighting of each node into account. It was introduced by Roy et al. [163]. Application specific measures can be easily incorporated to further specialize time-varying Fuzzy Contour Trees.

The *degree centrality* of a node is defined as the degree of the node, that is the number of connected edges. It hence records edge insertions and deletions in the sub-alignments.

The *betweenness centrality* of a node is the number of shortest paths going through it. Hence, using the betweenness centrality measure, the focus lies on structural changes in the sub-alignments.

Examples for the different coloring options are given in Figure 4.8.

Choosing Time Steps. Time steps can be selected via mouse click on the time selector in three different modes: *window selection*, *multi-selection* and *periodic selection* as a special case of multi-selection. See Figure 4.8 on the preceding page for examples. Toggling window and multi-selection is done via double click on a time step. Pressing control in multi-selection mode enters periodic selection.

In *window selection* (Figure 4.8a on the facing page), clicking on a time step selects this time step as the center of a time window of user-defined size. These windows are identical to the windows that are used by the coloring measures and pre-computed results from the measure calculation are re-used. The window selection is beneficial when a connected sub-interval of time steps is analyzed. Multiple adjacent selections of “sliding windows” provide insight in the development of the topological structure over time. New branches emerge with a low opacity, become more opaque as they become established and eventually fade (or stay). See Figure 4.14 on page 67 for an example.

The *multi-selection* (Figure 4.8c) is intended to provide deeper insight in individual members and their differences and similarities. By clicking on different time steps, they are added to the selection. The sub-alignment for the selected time steps is calculated and the time-varying Fuzzy Contour Tree is generated. There are no requirements to the selection, between one and all time steps can be selected.

As a special case of the multi-selection, the *periodic selection* (Figure 4.8b) assists the selection of periodic time steps. By providing the period, each selection is repeated for all cells with the given period. To easier detect periodic occurrences of branches in the data, markers that indicate the selected period are shown above the time selector while the periodic selection is activated. Periodic selection allows for example to select results for a single month over all available years as shown in Figure 4.12a on page 64.

Manipulating the Selection. In addition to the different selection modes, a given selection of time steps can be manipulated using the buttons under the time selector (Figure 4.6 on page 59). The left and right buttons shift the selection one time step to the left and right respectively. The spacing between multiple selected time steps remains the same if the overall time interval allows it. If however a selected time step would be shifted before the start or after the end of the data set, the selection remains untouched while other selected time steps are still manipulated, resulting in changed spacing.

The play button triggers an automated shifting of the selection to the right every second until either a selected time step reaches the end of the data set or the pause button is clicked.

4.4.2 Interaction with Time-Varying Fuzzy Contour Tree

Branch highlighting in the Fuzzy Contour Tree is unchanged. Hovering a branch highlights this branch and all its bundled edges and ancestors while all other branches are grayed out. At the same time, *member highlighting* is triggered. Here, time steps in the time selector that contain the highlighted branch are highlighted in the branch color, all others are grayed out. In contrast to the previous version, this highlighting not only takes place for time steps that are contained in the presented Fuzzy Contour Tree, but for all available time steps. Like this, navigating the data set and finding patterns in branch occurrences is facilitated. See Figure 4.12a on page 64 for an example.

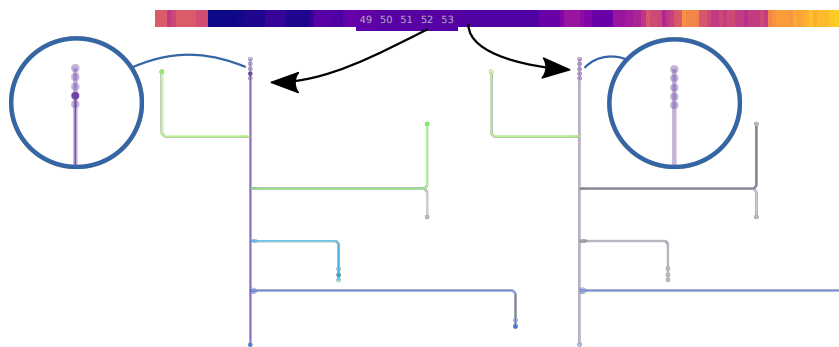


Fig. 4.9.: Tree highlighting: Left: hovering selected time steps highlights the complete individual contour tree. Right: for time steps that are not part of the current Fuzzy Contour Tree, only contained branches are highlighted without indication of specific nodes and saddles.

Hovering a time step in the time selector triggers *tree highlighting* of the corresponding individual contour tree in the Fuzzy Contour Tree. For selected time steps, the complete contour tree is contained in the shown Fuzzy Contour Tree and the individual contour tree with its nodes and saddles is indicated with thin lines. If the hovered time step is not selected and thus not part of the current sub-alignment, it is likely that some branches of the corresponding contour tree are missing in the current Fuzzy Contour Tree. Nonetheless, branches that are present are highlighted without marking specific saddles and leaves. See Figure 4.9 for both cases. The introduction of tree highlighting for time steps that are not selected allows relating these time steps to the current Fuzzy Contour Tree.

4.5 Results

To illustrate possible applications and the usefulness of our approach, we applied time-varying Fuzzy Contour Trees to different real world examples.

4.5.1 Sea Ice

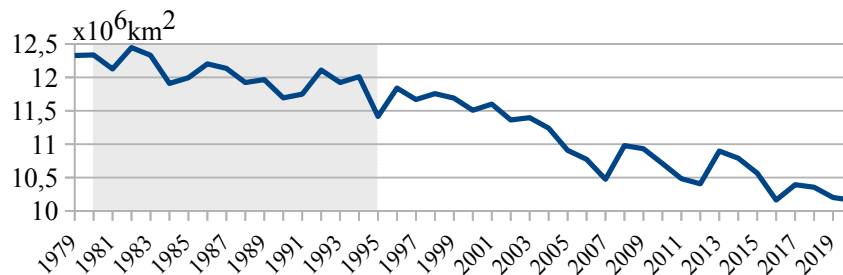


Fig. 4.10.: Annual mean sea ice extent in the arctic with indication of the analyzed time span [174].

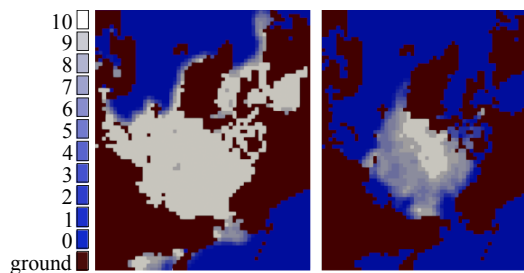


Fig. 4.11.: Arctic sea ice extent in January (left) and August 1994 (right).

The sea ice data set describes arctic sea ice concentrations and is provided by the National Snow and Ice Data Center [39]. The concentrations are given as tenths of grid square area covered by ice. In addition, we set the value -1 for grid cells that are not over sea. We applied time-varying Fuzzy Contour Trees to sea ice concentrations from 1980 to 1995 and applied a mild smoothing filter to the data to avoid non-binary contour trees. See Figure 4.10 for the mean arctic sea ice extent per year between 1979 and 2020. Plots of the sea ice extent in January and August 1994 are given in Figure 4.11.

Containing only discrete values between -1 and 10, this data set poses the following challenges:

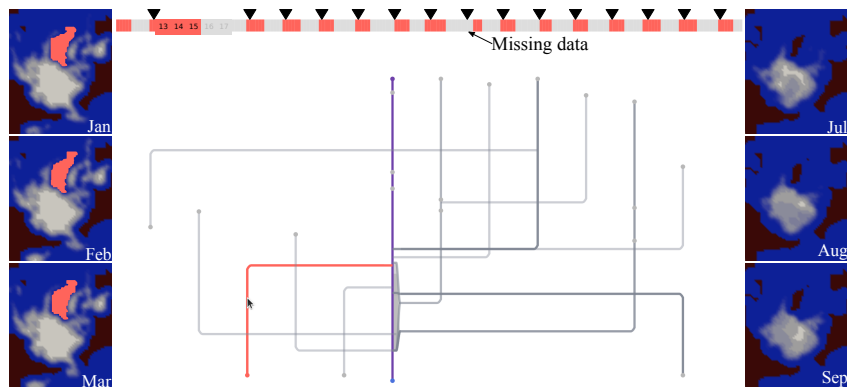
1. Large plateaus with constant values result in contour trees with branches of vanishing persistence. In this case, the alignment can not be executed using a metric that relies on persistence: creating a new node in the alignment instead

of matching two nodes is always the cheapest option with zero cost. Hence, no matching is performed.

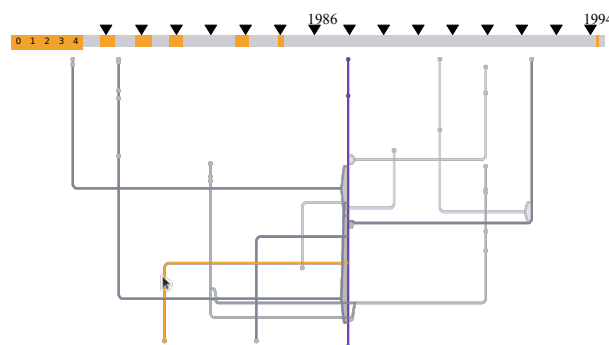
2. With saddles and extrema on discrete levels, finding a suitable layout for a contour tree is a challenge since leaves tend to be at the same height as saddles of other branches. This challenge aggravates for layouts of whole alignments.

Due to the flexibility of our system in back-end and front-end, the time-varying contour tree framework can still deal with this data set: the volume metric and the added overlap metric are possible choices that do not incorporate the persistence of branches. Both provide good matching results despite the branches with vanishing persistence. Optimized branch spacing tackles the challenge of discrete levels for saddles and extrema by shifting the branches to obtain a clearly laid out Fuzzy Contour Tree.

An exemplary application of time-varying contour trees is the analysis of the ongoing decline of the arctic sea ice extent.



(a) **Periodic behavior** in the sea ice data set. The marked branch is present only during winter months.



(b) **Vanishing branches over time** show the yearly decreasing extent of the sea ice. Around 1994 the sea ice extent peaked, which can be seen by the re-occurrence of the highlighted branch.

Fig. 4.12.: Insights using periodic selection: the periodic marker is set to 12 and the volume metric was used.

Figures 4.12a on the facing page and 4.12b on the preceding page show the time-varying Fuzzy Contour Tree for the sea ice data set with branches of typical behavior highlighted. **Comparing** the winter months of every selected year, the highlighted branch in Figure 4.12a can be found and it is easy to **separate** months where it occurs from months where it vanishes, and **combine** the respective branches in groups. In Figure 4.12b, the selected branch also occurs only during winter months. However, it vanishes in 1986, to re-occur in early 1994. Both of these behaviors, periodical occurrence over the whole data set and periodical occurrence only in about the first half of the considered time frame with potential re-occurrence between 1992 and 1994 can be seen in multiple branches.

The behavior in Figure 4.12a reflects the periodical increase of the sea ice during winter months over the whole considered period. The highlighted minimum (ocean) is only matched if it is surrounded by sufficiently large maxima (sea ice). Similarly, branches sharing the behavior of the highlighted branch in Figure 4.12b represent areas of the arctic sea that were frequently surrounded by sea ice during winters around 1985 but not any more.

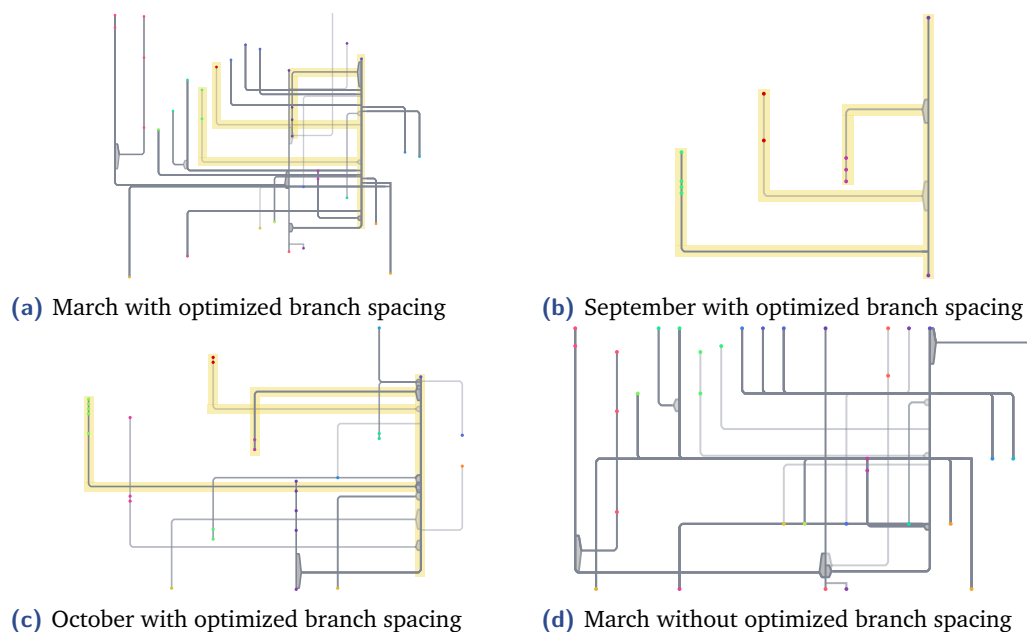


Fig. 4.13.: Sea ice extent during individual months. Similar structures are highlighted. For March, the comparison between the Fuzzy Contour Tree with (a) and without optimized branch spacing (d) is given.

Periodic selection allows the selection of individual months over all years. An example for the **comparison** of individual months from 1980 to 84 is given in Figure 4.13. Here, we used the overlap metric for the alignment and illustrate the usefulness of optimized branch spacing in discrete data sets. The much higher

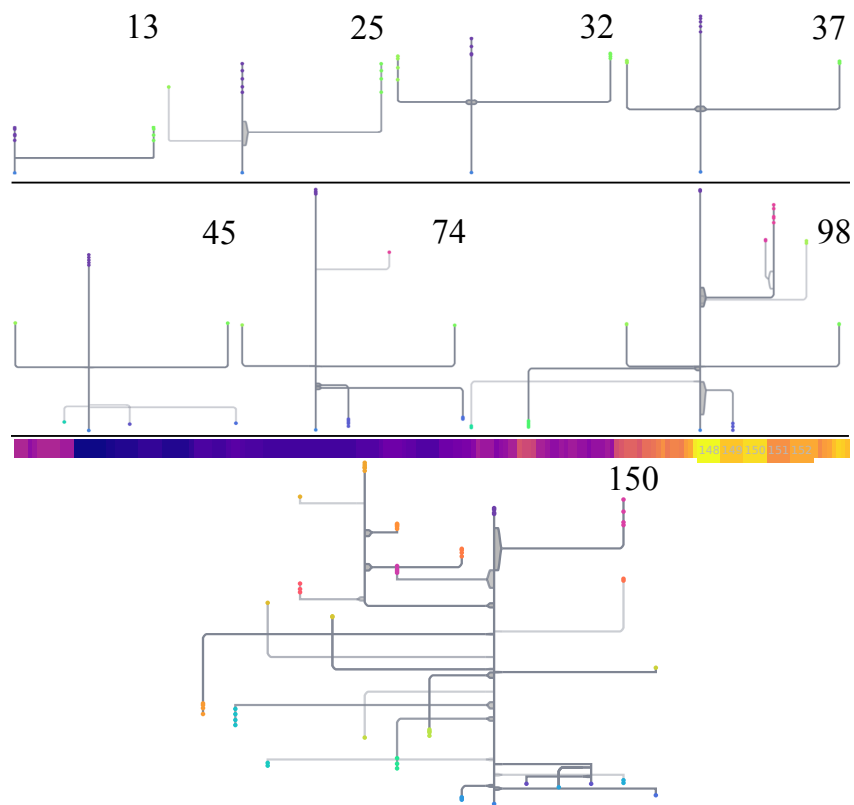
complexity of the main structures in the Fuzzy Contour Tree during winter months indicates the larger extent of sea ice in different ice floes. Although these floes are interrupted by islands and potentially even free to move, complicating a matching between time steps, similar structures are visible between the months, indicating areas of the arctic ocean that are covered by ice around the whole year.

4.5.2 Convection Simulation

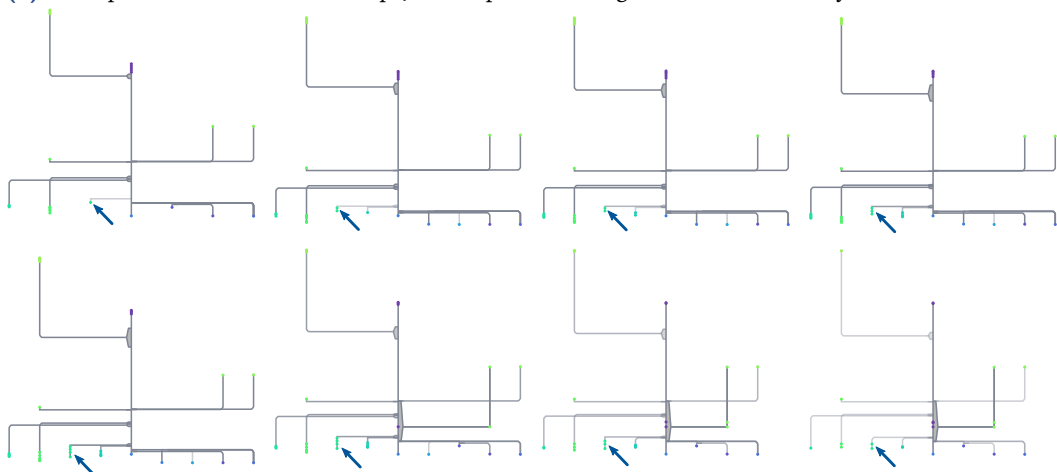
Ensemble members of the two-dimensional convection simulation described in Section 3.4.3 on page 43 are also simulated over time. Material at rest is heated around the pole, begins to rise, and forms a plume. This behavior is clearly reflected in the time-varying Fuzzy Contour Trees in Figure 4.14a on the facing page. Advancing the selected window step by step through the data provides an overview of the topological structure at individual time steps but also their connection and development: Until time step 74, the pole is heating up. Then, the plume forms and more and more temperature minima are enclosed by the formed plume. The time selector color shows the degree values, reflecting the dynamic behavior of the plume towards the end. I chose representative time steps and scaled them to highlight the raising temperature; screenshots of a sliding window for the time steps between 5 and 12 are given in Figure 4.14b on the next page, showing appearance and vanishing of branches when advancing the selected window time step by time step.

4.5.3 Cloud Top Pressure

While time-varying Fuzzy Contour Trees proved to be beneficial to analyze the topological behavior of climate related time-dependent scalar fields, they inherit the limitations of Fuzzy Contour Trees and contour trees. I previously gave an example for the necessity of similar topological structures in Section 3.4.4 on page 47. By implementing the overlap metric, this requirement was weakened. However, by its nature as an augmented contour tree, the Fuzzy Contour Tree tends to become cluttered with a large number of branches and leaves. As an example for this, I apply time-varying Fuzzy Contour Trees to the cloud top pressure field of simulation results for the weather over central Europe in Figure 4.15 on page 68 (subset of the HD(CP)² data set [173]). Already contour trees for individual time steps of this data set pose a visualization challenge because of their complexity. Although the matching of different extrema over time works reliably, the resulting Fuzzy Contour Tree is difficult to analyze, especially in a static image. Branch and member



(a) An expressive subset of time steps; the step number is given above the Fuzzy Contour Trees.



(b) Advancing time step by time step, the evolution of branches gets visible between time steps 5 and 12. An example is the marked branch: it starts with a low frequency, over time more and more members accrue and the frequency rises, then it declines again.

Fig. 4.14.: Sliding window: advancing through the convection simulation result provides a clear understanding of the ongoing processes. The given step numbers refer to the center of the selected window of width 5.

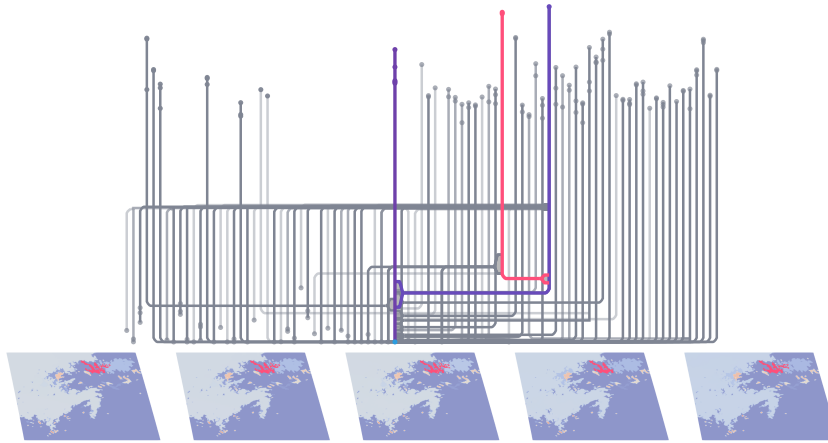


Fig. 4.15.: Limitations of time-varying Fuzzy Contour Trees: The time-varying Fuzzy Contour Tree for the first 5 time steps of the cloud top pressure in the HD(CP)² data set and an example for matched extrema. The branch corresponding to the shown components is highlighted in the Fuzzy Contour Tree.

highlighting as well as optimized branch spacing prove beneficial also in this case and structures are visible despite the high amount of nodes and branches.

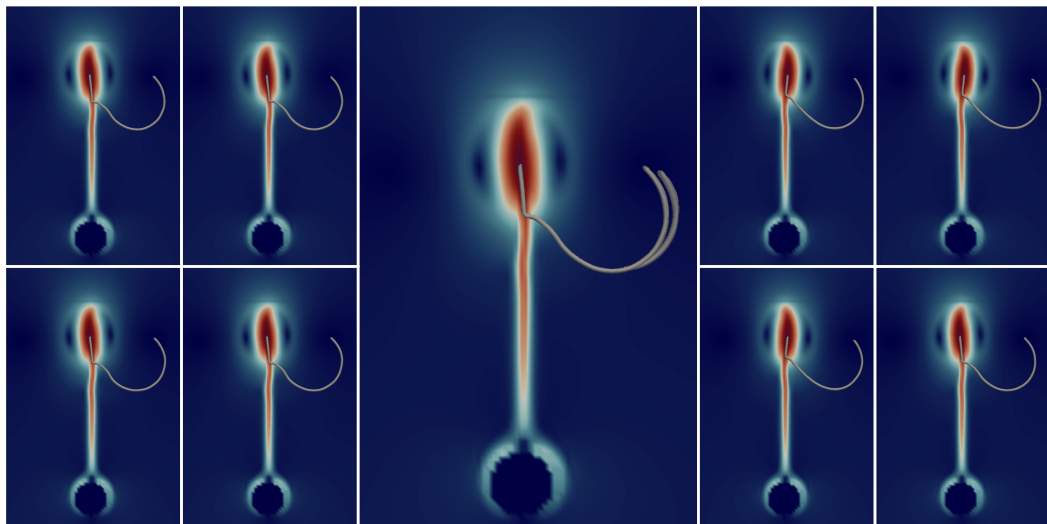
4.6 Conclusion

Applying Fuzzy Contour Trees in the time-varying setting, we are able to provide a holistic view on time-dependent scalar fields. We adapted back- and front-end to the specific challenges of time-dependent data visualization, resulting in the time-varying Fuzzy Contour Tree interface. This interface allows **comparison**, **combination** and **separation** of arbitrary subsets of time steps based on their topological structure, without requiring them to be consecutive.

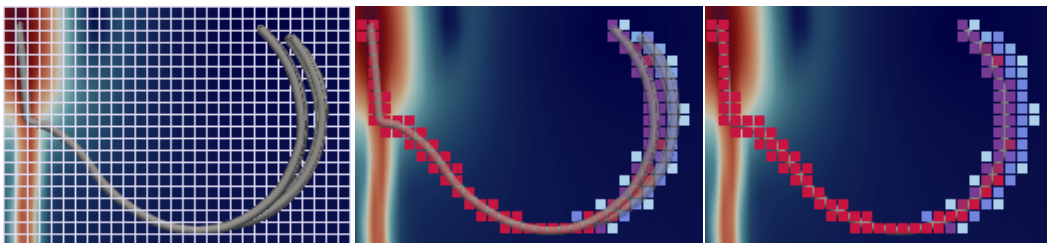
In addition to further research opportunities on Fuzzy Contour Trees (Section 3.6 on page 49), existing approaches to automatically detect application specific patterns or features could be used to augment the time-varying Fuzzy Contour Tree. Including this information could for example allow to highlight specific time steps in the time slider or to propose subsets for comparison.

In summary, extending and applying Fuzzy Contour Trees to time-varying climate data results in useful, coherent visualizations, providing a more general overview of the data than common analysis approaches.

Uncertain Trajectories in Vector Field Ensembles



(a) A specific streamline is integrated in all members. Center: superposition of all member streamlines.



(b) Detail of (a): counting the member streamlines that pass by each cell, the visitation map is created.

Fig. 5.1.: Visitation Maps show the transport behavior in two dimensional vector field ensembles.

Moving from scalar to vector fields, field lines are an effective approach to visualize the flow of fluid particles. A straight forward and easy to use generalization of these techniques to ensembles are *visitation maps*, used to elucidate the transport behavior that is described by an ensemble of two-dimensional vector fields (Figure 5.1). Using this visualization technique, it is possible to **compare** the behavior of the whole ensemble to the desired outcome and more likely paths to unlikely paths. It is furthermore possible to identify common paths of ensemble members, **combine** them and identify areas with similar flow behavior. Also, it is possible to **separate**

areas of possible impact from ones where impact is likely and ones where it is impossible. Furthermore, outlier and areas with different flow behavior can be separated.

The computation of visitation maps is in practice simple and, given an initial distribution, utilizes Monte Carlo sampling of trajectories across a vector field ensemble. For large (or even medium sized) vector field ensembles, however, the naïve approach becomes prohibitively costly: every time the initial distribution is changed, the computation restarts from scratch with high running times. Moreover, all ensemble members are required for sampling. For large ensembles, storing all members is often impossible and in situ analysis is required, forcing the user to fix the initial distribution at data creation time, without the possibility to change it afterwards. Hence, the straightforward approach is not adequate for interactive exploration of uncertain vector fields.

Together with Prof.Dr. Christoph Garth, I developed *Visitation Graphs* as a novel intermediate representation of the flow behavior in an ensemble of two-dimensional vector fields and published it at iPMVM 2020 [*117]. Representing flow fields as Visitation Graphs (treated in Section 5.2.1), the stored data is tailored to fast generation of visitation maps from arbitrary initial distributions as explained in Section 5.2.2. This enables the interactive exploration of vector field ensembles where the naïve approach fails. Furthermore, Visitation Graphs are a data reduction method that outperforms downsampling in terms of information loss and that can be calculated in situ (Section 5.2.3).

5.1 Visualization of Uncertain Vector Fields: Background and Related Work

Originating in experimental visualization techniques and observations from nature, field lines are an intuitive method to visualize fluid flow.

Streamlines are at every point tangential to the vector field, showing the direction a massless particle will follow at any point in time. Mathematically, they are defined by $\frac{dx}{dt} = \vec{v}(t, x)$ with $x(t_0) = x_0$. An example for streamlines is the visualization of a member in the two-dimensional convection simulation in Figure 5.1 on the preceding page. Streamlines in the three-dimensional convection simulation are shown on the right in Figure 3.24 on page 43.

Streaklines can be observed experimentally by steadily injecting dye at a fixed point in the fluid. Hence, they follow the locations of particles that are seeded at a fixed point and then further influenced by the flow over time. An example is given on the left in Figure 3.24 on page 43.

Pathlines are trajectories of individual fluid particles over time. For a static vector field, they conform to streamlines.

Timelines are the lines formed by simultaneously moving particles that are seeded in a seed curve at a previous instant in time.

All these field lines can be calculated forward or backward in time. Being the fundamental visualization tool for flow fields, a plethora of applications and variations of field lines exist [131].

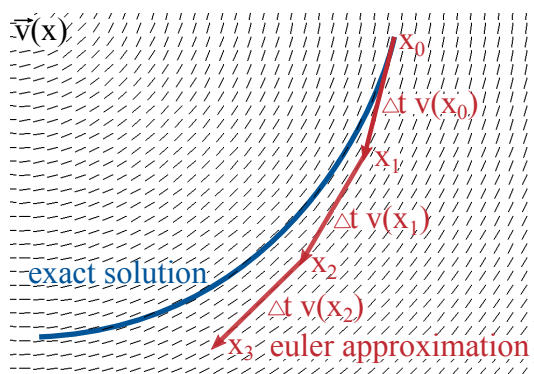


Fig. 5.2.: The **Euler method** is a simple approach to approximate streamlines.

As results of differential equations, an exact calculation of field lines is impossible in general. Thus, they are approximated using appropriate integration schemes, for example from the family of Runge-Kutta methods for streamlines. The simplest method in this family of iterative methods is the Euler method, illustrated in Figure 5.2. Step by step, the vector at the current position is evaluated and traced for a user defined time Δt . In the attained position, the vector is again evaluated and traced. This is repeated until the streamline becomes local or reaches a maximum length. Thus, in step n of the Euler scheme, the position x_{n+1} in the (time-dependent) vector field $v(t, x)$ is obtained as

$$x_{n+1} := x_n + \Delta t v(t_n, x_n).$$

Higher order Runge-Kutta methods improve the encountered approximation error of $\mathcal{O}(\Delta t^2)$ via insertion of additional sampling points compared to the Euler scheme. Furthermore, adaptive methods vary Δt based on the underlying field.

Different approaches for the approximation of a longer particle trajectory by a sequence of (certain) flow maps have been examined. Agranovsky et al. give a

two phase approach extracting a basis of known pathlines in situ and calculating arbitrary integral curves post hoc from the extracted results [1]. I introduce two similar phases, carrying the idea to approximate longer curves using smaller ones over to visitation maps.

Uncertain Flow Visualization. Dealing with uncertain vector fields, and thus with varying field lines for every member, many visualization approaches have been proposed. Examples in vector field ensembles are: showing ensemble members vanishing over time [45], enabling the user to compare single members to the whole ensemble using glyphs [165] and summarizing ensemble members while highlighting outliers and median in Contour/Curve Box Plots [133, 209]. The topology of ensembles in two and three dimensions was determined by Otto et al. [149, 150]. Hummel et al. gave a comparative visual analysis for ensembles of time-varying vector fields using a Lagrangian framework [87]. A two dimensional comparative visual analysis was presented by Jarema et al. [93].

Uncertainty arising from interpolation and prediction of missing measurements was treated using tubes of varying size [21], glyphs, and parallel coordinates for magnetic resonance spectroscopy data [59, 60], and flow radar glyphs for time-dependent vector fields with uncertainty given as an interval [82].

Random fields are a stochastic uncertainty model. Following the approach of in situ data reduction by summarizing statistics of certain properties, random fields frequently arise. While in mathematics, the generalization of stochastic processes to higher dimensions is called *random field*, different names have been used in the visualization community up to now. Otto et al. speak of *uncertain vector fields* [150], Ferstl et al. use the term *ensemble of vector fields* [61], Sevilla-Lara et al. speak of *distribution fields* in computer vision [175] and Love et al. use the more general term *spatial multivalued data* [122].

Visitation Maps. A highly intuitive and established generalization of streamlines to vector field ensembles and random fields are *visitation maps*. Previous authors have defined visitation maps in an ad hoc manner as the empirical distribution within each cell in the domain obtained as the percentage of generated trajectories of a given length and with a given start point passing through the cell (Figure 5.1 on page 69)[28, 97, 98].

However, the resulting ad hoc calculation renders the use of visitation maps challenging due to computational effort inherent in their numerical approximation. Given

an initial condition (i.e. starting location or initial distribution), the visitation map is typically estimated through direct sampling. For a faithful approximation, a high number of samples is required. For small data sets, parallel computation using for example GPUs can be leveraged to achieve interactive re-computation upon modification of the initial condition. For example, Bürger et al. demonstrated an interactive visualization of ensemble vector fields with visitation maps using GPU-based Monte-Carlo particle tracing [28]. However, for larger data sets, trajectory computation is a difficult problem and has to be handled through non-interactive out-of-core techniques [145] or parallel algorithms [158]. Following the naïve approach of visitation map calculation, the expensive calculations need to be re-done from scratch, every time the initial condition is changed. Also, the complete data is required for sampling.

A recent variation of visitation maps was proposed by Ferstl et al. as streamline variability plots [61]. In contrast to visitation maps, variability plots are generated by projecting confidence ellipses for obtained streamline clusters in PCA space to domain space. This yields an envelope for most obtained streamlines, together with a calculated mean streamline. For large data sets, the challenge to integrate large numbers of streamlines remains the same.

Representing Vector Fields as Graphs and Webs. To make interactive visitation maps available for larger ensembles, we developed a graph representation for flow field ensembles that provides the ideal starting point for visitation map generation – the Visitation Graph.

Representing information on a given vector field in a graph structure was earlier considered in the forms of Flow Graphs by Nouanesengsy et al. [145] and as Flow Webs by Xu et al. [215]. The Flow Graph contains a node for each block in an underlying grid, and each two neighboring blocks are connected via a weighted edge. The weight of the edge connecting two blocks is determined as the probability that a seeded particle in one of the blocks is transported to the other one by the flow. Afterwards, this flow graph can be used to estimate the workload for each block in parallel streamline calculation.

While Flow Graphs are an efficient approach to estimate workload, they are not suitable for visitation map approximation, as visitation maps calculated from such graphs suffer from the lack of variety in possible directions in the graph. In a Flow Graph each cell is connected only to its four direct neighbors, which results in an extremely coarse approximation of the visitation map. An example for a visitation

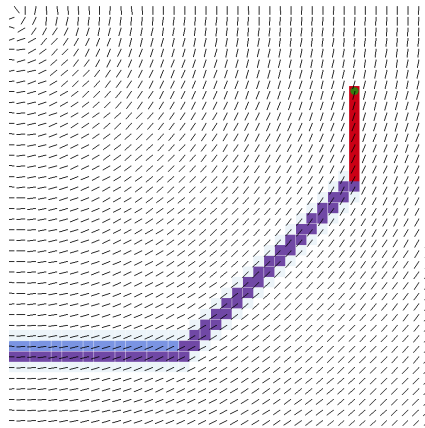


Fig. 5.3.: Visitation map generated from a flow graph recording connections from each cell to its 8 neighbors, weighted with the corresponding probability that a streamline will continue in the cell. Comparing to the mean vectors of the underlying vector field, the result is a very coarse approximation to the actual visitation map.

map based on a graph where each cell is connected to its eight neighbors can be found in Figure 5.3.

In Flow Webs, regular axis-aligned sub-regions of the given domain represent nodes in the graph. By sampling streamlines backwards through the regions, links and weights between different sub-regions are determined. While Xu et al. do not consider uncertainty, our approach can be viewed as an adaptation of the Flow Web concept to ensembles to create visitation maps, based on a generalization of the Flow Connection Matrix (adjacency matrix of the Flow Web).

As opposed to the Flow Graphs by Nouanesengsy et al., Flow Graphs by Ma et al. [126] are a tool for streamline and pathline exploration of three-dimensional flow fields. Here, nodes in the graph do not only represent spatial regions but also streamlines in the field. Edges between nodes are assigned various interpretations depending on the kind of nodes they interconnect.

Adaptive transition graphs for vector fields have been employed by Szymczak to calculate Morse connection graphs for piece wise constant vector fields on surfaces [190]. His method of refining transition graphs in an adaptive manner according to strongly connected components can be interpreted as perturbing the vector field near trivial Morse sets to remove recurrent features in the resulting Morse connection graph.

Multiple approaches employing graphs to track and visualize states and state transitions in time-varying vector fields have been proposed, for example by Gu and Wang

[73] and by Jänicke and Scheuermann [92]. A recent survey on the utilization of graphs in visualization was given by Wang et al. [201].

In Situ Analysis and Visualization. In situ analysis, i.e. data analysis taking place at creation time while the data is still in memory, is an attractive possibility to handle data sets that are too large to store. In addition, slow data output is avoided and data can be pre-processed at creation time. As computational power increases and thus simulations of massive ensembles become common, the need for effective in situ processing is ever more pressing.

Haines, forced by (in 1994) huge data sets of “10s to 100s of Gigabytes” developed an early in situ visualization for large unsteady data sets [75]. Interfaces for leading visualization libraries like Paraview [55] and VisIt [114, 210] were developed, allowing in situ analysis and visualization of simulations. Yu et al. developed parallel in situ visualization of volumes and particles resulting from the simulation of combustion engines [218].

While in situ visualization provides crucial insights in the simulation behavior at runtime, interactivity and thus exploration of the data is rarely possible. Pre-processing simulation data in situ and benefiting from reduced data in flexible and scalable post hoc analysis is a possibility to combine the advantages of in situ and post-processing. This new paradigm was for example illustrated by Childs [44]. Several approaches were already presented that follow this paradigm. For example Lakshminarasimhan et al. proposed ISABELA for in situ sorting and error bounded compression [107].

ISABELA was developed further by Lehmann et al. for the specific task of visual exploration of scientific simulations [112]. Wang et al. base their in situ reduction and following rendering of largescale time-varying, multivariate data sets on expert knowledge [202]. Sampling was used for in situ reduction by Woodring et al. to provide an interactive visualization of cosmology data [213]. An in situ image based approach with post analysis of features was proposed by Ahrens et al. [2].

An other approach to reduce data in situ is to summarize statistics of properties of interest during the simulation. Afterwards, suitable visualization approaches are used for data exploration. This was done recently by Dutta et al. [53]. A survey was given by Li et al. grouping current research on in situ data reduction in a spectrum between lossless and very lossy techniques [113].

Following the in situ, post-processing combination by Childs [44], our approach processes steady two-dimensional vector field ensembles in situ to construct a

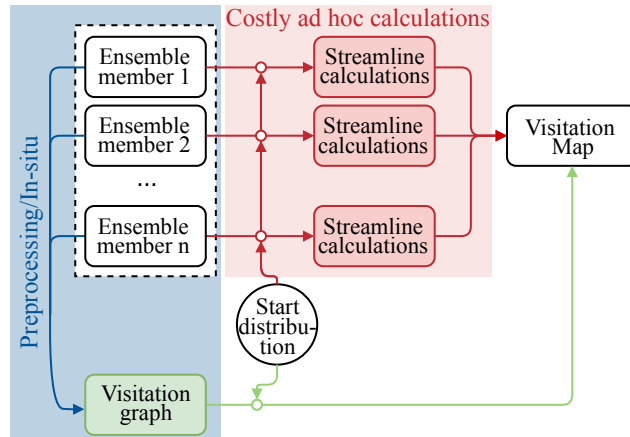


Fig. 5.4.: Comparison of a visitation map creation in the ad hoc way and using Visitation Graphs: the former requires costly ad hoc streamline computations for every new start point. After pre-processing, the latter is able to generate visitation maps interactively requiring only simple matrix operations.

Visitation Graph. With an upper bound for storage requirements that does not depend on the number of ensemble members, this Visitation Graph is an ideal structure to summarize ensembles. Furthermore, the Visitation Graph resolution can be chosen freely and does not have to mirror the resolution of ensemble members. This provides a means for data reduction, tailored to the task of creating visitation maps for interactive exploration of ensembles potentially too large to store. Based on the Visitation Graph, an interactive post hoc exploration of the ensemble using visitation maps is possible once the simulation and pre-processing are complete.

5.2 Visitation Graphs: Interactive Ensemble Visualization with Visitation Maps

Our research considers two-dimensional vector field ensembles or random vector fields. These are partitioned into cells of their domain of definition $\mathcal{C} = \{c_1, \dots, c_M\}$. A visitation map \mathcal{V} represents for each c_j the distribution of streamline samples of integration length $T > 0$ that “hit” the cell c_j , i.e.

$$\mathcal{V}(c_j) := P(S(X, T) \cap c_j \neq \emptyset),$$

where $S(X, T)$ is the set of all points of the streamline sample, and its seed point X is a random variable with a fixed initial distribution (in traditional visitation maps

X is a uniform distribution in a single cell). In practice, this probability is computed through Monte Carlo sampling as an average over a set of streamline samples.

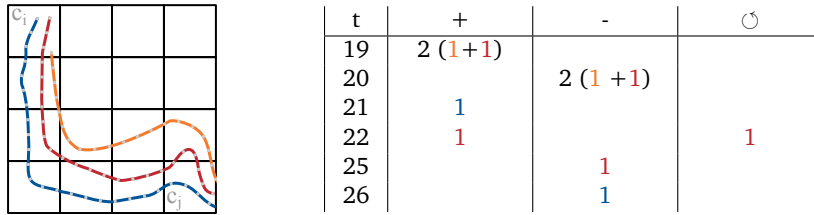
To speed up visitation map generation, our approach performs the sampling calculations in a pre-processing step that can be executed in situ, and stores the result as a compact *Visitation Graph* from which visitation maps can be derived quickly in a later state. Thus, we avoid the problem of ensembles being too large to handle interactively. In addition, a tradeoff between pre-processing time, storage requirements, and visitation map accuracy can be made.

We define the Visitation Graph as a directed graph $G(V, E)$ whose nodes V are the cells of the partition, i.e. $V = \mathcal{C}$, and E contains an edge between c_i and c_j if and only if a streamline sample of length T' starting in c_i hits c_j during integration. For each edge (c_i, c_j) , streamlines starting in cell c_i are considered. Their (re-)entry and exit in cell c_j are recorded.

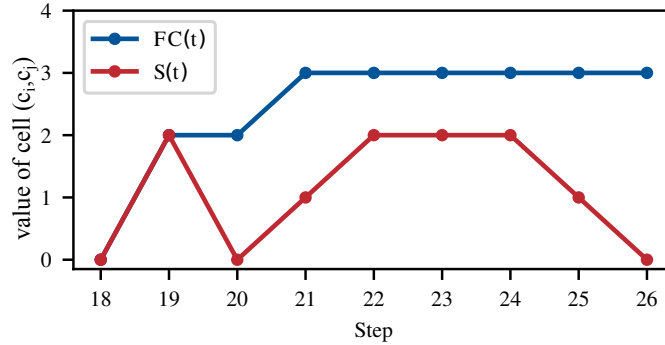
Streamline samples are approximated as a polyline with k points, originating in every cell c_i . In practice, we (re-)use the computational grid of the ensemble as the partition \mathcal{C} , and approximate streamlines using a second-order Runge-Kutta scheme with fixed step size (hence k steps of step size Δt). Both of these aspects of the technique are easily modified.

Information is collected similar to the Flow Connection Matrix [215]. However, the Visitation Graph keeps track of more information than just the number of passing streamlines, namely the events and time points.

From the Visitation Graph, visitation maps can be assembled. This eschews several problems inherent in the ad hoc calculation. A comparison of the two approaches can be found in Figure 5.4 on the facing page. To approximate a visitation map of length T , it is not required that $k\Delta t := T' = T$ holds; in practice, streamline samples can be much shorter than the desired visitation map length, at the cost of accuracy. This allows to trade off accuracy and speed in visitation map creation for speed in Visitation Graph creation and storage size of the Visitation Graph, since shorter streamlines yield less events that need to be stored in the graph. Furthermore, the Visitation Graph resolution is not necessarily the same as the resolution of ensemble members. This provides a method for data reduction tailored to the task of creating visitation maps for interactive exploration of ensembles potentially too big to store. See Section 5.2.3 on page 84 for more details on data reduction using Visitation Graphs.



(a) **Visitation Graph creation:** streamlines starting in cell c_i that pass by c_j are considered and all events are tracked with time point. Table entries are colored according to the triggering streamline.



(b) Values of the **Flow Connection Matrix and snapshot** for (a), omitting normalization. $FC(t)$: number of streamlines that passed by, $S(t)$: number of streamlines that are currently in the cell.

Fig. 5.5.: Visitation Graph Creation

5.2.1 Efficient Computation of Visitation Graphs

During creation each cell is considered once as start cell of stream lines. Cells c_i and c_j are connected in the Visitation Graph if a streamline of a given length starting in cell c_i passes by cell c_j . For each cell c_j that is connected to the considered start cell c_i in the Visitation Graph, the following events of interest are recorded for each step during creation:

- (+) how many streamlines starting in cell c_i enter cell c_j ,
- (-) how many streamlines starting in cell c_i leave cell c_j , and
- (⊙) how many streamlines starting in cell c_i re-visit cell c_j

The last entry is obtained by keeping track for each cell whether the considered streamline has already entered c_j or not. An illustration of Visitation Graph generation can be found in Figure 5.5.

During Visitation Graph creation, the number of randomly seeded streamlines started in each cell can be determined either based on the size of the cell ([215]) or on the estimated contribution to the final Visitation Graph. While the first option is straight forward, the latter is achieved as follows: the number of cells n_h that have

been hit by any streamline that was integrated from the considered start cell is determined and a size for streamline bunches is set as parameter s_b . Every time the integration of a bunch of streamlines is completed, it is decided whether one more will be started or not using

$$\frac{n_h}{s} < \frac{1.0}{s_b}. \quad (5.1)$$

Where s is the total number of integrated streamlines from the considered start cell. If expression 5.1 evaluates to false, the next bunch of streamlines is integrated, else the Visitation Graph creation for the considered cell stops. This stop criterion can be interpreted as follows: it is unlikely that a streamline from the upcoming bunch of streamlines passes by a cell that was not passed before since the number of passed cells per integrated streamline is smaller than one out of a bunch of streamlines.

Doing these calculations for different ensemble members is independent. Hence, it can be done in situ. For every ensemble member, entries in the Visitation Graph are updated or added. While the Visitation Graph can be built completely in situ, it does not have to. In case of ensemble members occupying most of the available memory storage, it might be impossible to create the whole Visitation Graph simultaneously. In this case, single members or even single cells can be processed in situ and then be stored. In a post processing step all partial results are combined to the final Visitation Graph. Depending on the available memory, every possible storing frequency in between “storing for each cell or member” and “storing once at the end” can be used providing a tradeoff between postprocessing time and memory requirements.

5.2.2 Efficient Approximation of Visitation Maps from Visitation Graphs

Given an initial distribution of cells or a single start cell, Visitation Graphs are an optimal base to calculate visitation maps.

Using the Visitation Graph consisting of M nodes with tracked events of interest, two different matrices can be generated for every time point $t \leq T'$ (integration length): The *Flow Connection Matrix* $FC(t)$ as described by Xu et al. [215] and the *snapshot* $S(t)$. The Flow Connection Matrix is an $M \times M$ matrix given by the adjacency matrix of the Visitation Graph where the weight of edge (c_i, c_j) is given by the probability that a streamline starting in cell c_i passes by cell c_j before time point t . The entries are calculated as the sum of the differences of first and third entry of the stored 3-tuples until t , divided by the total number of integrated streamlines starting in c_i . While the first entry provides the information how many streamlines

from c_i have entered c_j , the third entry ensures that re-entering streamlines are not counted twice. Note that Xu et al. generated the Flow Connection Matrix focusing on infinitely many streamline steps, thus our approach constitutes a generalization of their graph structure.

Entry (c_i, c_j) of the $M \times M$ dimensional *snapshot* matrix $S(t)$ specifies the probability that particles starting in c_i are in c_j after integration time t . It is calculated as the sum of the difference of first and second entry of the stored 3-tuples until t , divided by the total number of streamlines starting in c_j . Thus every time a streamline enters c_j , the snapshot entry increases by one and as the streamline leaves c_j , the snapshot entry decreases by one. This is done regardless of re-visiting. Examples for the calculation can be found in Figure 5.5b on page 78.

The total number of streamlines is required to generate Flow Connection or snapshot matrices from the Visitation Graph. This number varies between cells since it depends either on their estimated contribution or the cell size, as described in Section 5.2.1 on page 78. However, it is not necessary to store the total number of streamlines separately, since every streamline is recorded as entering in the start cell at the initial time point. Thus the total number of streamlines per start cell is available in the Visitation Graph.

Having an integration length T' that is independent of the final visitation map length T , the desired visitation map might have more or less steps than have been integrated while pre-processing. Being able to generate a visitation map based on streamlines that are shorter than the final map provides a tradeoff between accuracy on one side and shorter pre-processing time and storage savings on the other side.

Two scenarios for visitation map creation exist: Either $T \leq T'$, that is the final visitation maps has less steps than have been integrated in pre-processing or $T > T'$, that is the desired visitation map has more steps than have been calculated for the Visitation Graph.

If $T \leq T'$, the visitation map starting in cell c_i is given by the row representing c_i in the Flow Connection Matrix: the exact visitation map value $\mathcal{V}(c_j)$ is given in entry (c_i, c_j) of $FC(T)$. Considering possibly multiple start cells with a start distribution, a M -dimensional start vector v_0 is created holding the start probability for each start cell. This vector is then multiplied with $FC(T)$. Each entry of the resulting vector represents one cell in the grid and holds the visitation map values. The resulting visitation map gives the identical result as a traditional visitation map based on ad hoc created streamlines advancing for time T .

If, on the other hand $\mathbf{T} > \mathbf{T}'$, the desired visitation map length exceeds the number of pre-processing steps, the visitation map is not given in the Flow Connection Matrix but needs to be “assembled” from multiple $FC(t_i)$ where $\sum_i t_i = T$.

The law of total probability states that if $\cup B_i = \Omega$ is a countable partition of the entire sample space, then for any event A it holds:

$$P(A) = \sum_i P(A | B_i)P(B_i) \quad (5.2)$$

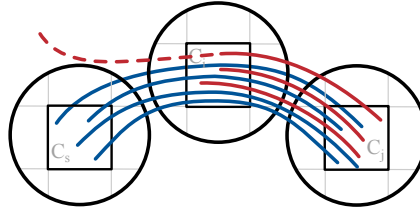


Fig. 5.6.: Probability approximation: the probability of a streamline path $c_s \rightarrow c_i \rightarrow c_j$ in time T while in c_i after T' is approximated by the probability of the streamline path $c_i \rightarrow c_j$ in $T - T'$. This leads in general to overestimation since every streamline path $c_s \rightarrow c_i \rightarrow c_j$ is also a path $c_i \rightarrow c_j$, but not all streamlines from c_i to c_j pass by (or start in) c_s before entering c_i (for example the dashed streamline).

For clarity of notation let in the following S , A , and B be arbitrary cells in \mathcal{C} . Considering the partition consisting of events

$A_t^S :=$ After time t , the considered streamline starting in cell S ends in cell A ,

for the fixed integration time in pre-processing T' , the law of total probability gives

$$P(B_T^S) = \sum_{A \in \mathcal{C}} P(B_T^S | A_{T'}^S)P(A_{T'}^S) \quad (5.3)$$

Where the event B_T^S is an event considering streamlines after time $T > T'$, thus no calculations for this event have been performed during pre-processing. To estimate $P(B_T^S)$, the conditional probability $P(B_T^S | A_{T'}^S)$ is approximated using $P(B_{T-T'}^A)$ in equation 5.3. So the probability that the considered streamline that starts in S ends in B after time T while ending in A after time $T' < T$ is approximated by the probability that a considered streamline starting in A ends in B after $T - T'$ steps. See Figure 5.6 for further explanation. This gives

$$P(B_T^S) \approx \sum_{A \in \mathcal{C}} P(B_{T-T'}^A)P(A_{T'}^S). \quad (5.4)$$

Since $P(B_{T-T'}^A)$ considers an integration time strictly smaller than T , the considered number of steps is potentially smaller than the number of pre-processing steps until time T' such that pre-processed results are available. If however $T - T' > T'$, the same approximation is repeated until the integration time is smaller or equal to T' .

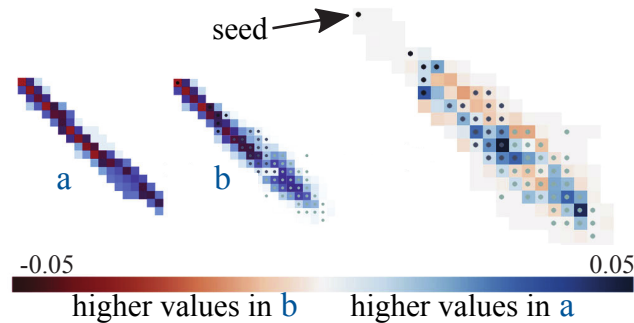


Fig. 5.7.: Comparison of traditional visitation map **a**, the approximation based on Visitation Graphs **b** and the difference between both. $T' = 50$, $T = 160$. Nonzero entries of the start vectors v_i are marked from black to light grey for increasing i . Before the first approximation step, the results are identical.

This approximation in general overestimates $P(B_T^S | A_{T'}^S)$. Equality is only given if $P(A_{T'}^S) = 1$ since then $P(B_{T-T'}^A)$ is independent of the point in cell A reached by the streamline from S in time T' . Thus every approximation step will induce an error in the resulting visitation map (cf. Figure 5.7).

We implemented this theoretical result as follows: the exact visitation map after time T' is calculated as described above using the start vector v_0 . Having reached the maximal calculated step, a new start vector $v_1 = v_0 \cdot S(T')$ is calculated. Using this new start vector holding the positions and probabilities for particles starting in the considered start cell S under the considered start distribution after T' steps, the proceeding visitation map is calculated using $FC(T - T')$. The results are combined ensuring to not have doubled results in nonzero entries of v_1 . This is repeated m times until $T - mT' \leq T'$.

Illustratively, the visitation map is followed as far as it was pre-computed, then new visitation maps are started at the end incrementally until the desired number of steps is reached (Figure 5.8 on the next page).

While a smaller number of pre-processing steps induces errors in calculated visitation maps, pre-processing time becomes shorter and storage requirements of the resulting Visitation Graph become smaller. This tradeoff is illustrated by experiments in Section 5.3 on page 88. Visitation map creation on the other hand becomes more expensive the more snapshots and Flow Connection Matrices are evaluated, thus

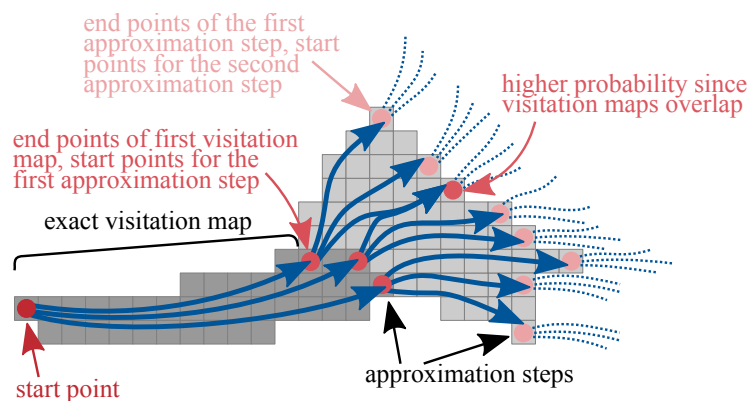


Fig. 5.8.: Illustration of our **visitation map approximation**: the first part of the generated visitation map is the exact visitation map generated from the data up to the streamline length T' in the Visitation Graph. Using snapshot and Flow Connection Matrix, the cells holding the particles after T' steps are determined. Using the resulting distribution as start distribution, again the exact visitation map is determined and attached to the first one and so on, until T steps have been reached.

the calculation time for visitation maps from the Visitation Graph decreases with increasing pre-processing integration time T' . In general, calculation times can be reduced using parallelization: Integrating streamlines from every cell to determine edges in the Visitation Graph can be heavily parallelized, since considering different cells is completely independent. If the ensemble members reside on different nodes of a cluster, these members can be processed independently on their nodes. The final Visitation Graph can then be assembled using the parallel reduction approach resulting in logarithmic time savings.

In an informal experiment, we observed nearly ideal speedup (7.96x for 8 CPUs) for streamline integration. The assembly of 10 partial results was executed on 2 CPUs with a speedup of 1.2. In addition, the calculation of Flow Connection and snapshot matrices for visitation map creation can be naturally parallelized for every entry in the resulting matrix. While these matrices are potentially of very high rank, the fact that entries of $FC(t)$ and $S(t)$ can be calculated independently can be used to generate only rows that are required for the computation (that is the rows whose entries in the start vector are nonzero) and save them in a sparse format. Especially for the first steps in visitation map creation, where the start vector contains only the chosen start cells, the number of needed rows is very small compared to the total number of rows.

5.2.3 Space Requirements and Data Reduction

Visitation Graphs are an optimal way to store, and potentially reduce, data in order to explore the data with visitation maps later on. The maximal storage requirements of the Visitation Graph representation are determined by the ensemble resolution $m \times n$ and the integration time T' . As such, they are independent of the number of ensemble members. An upper bound for the number of outgoing edges is $m \cdot n$, which is far from being tight. In reality, the number of outgoing edges is much smaller since it is impossible that every cell in the grid is passed by streamlines from every other cell. Also, edges are only possible between cells that can reach one another with a streamline of set length, thus only cells in a given, field dependent radius around each cell can be connected. At most $T' \cdot 4$ numbers need to be stored per edge. This upper bound will not be reached in realistic examples and most of the information stored for edges will contain far less than T' tuples (see Section 5.4 on page 91 for actual numbers in application examples). Thus, facing ensembles with a high number of members compared to the number of cells per ensemble, the Visitation Graph will save space without any additional data reduction. For other ensembles, multiple options leading to data reduction are available:

Visitation Graph Resolution. The Visitation Graph resolution is independent of the original resolution of the ensemble. Thus, a coarser resolution can be chosen to achieve data reduction. While a coarse Visitation Graph contains only nodes corresponding to a coarser grid, it is based on streamlines that are created based on the ensemble in its original resolution. Carrying over the information from the fine grid to the coarse one, coarse Visitation Graphs are superior to simple downsampling of the grid. See experimental results in Section 5.3 on page 88 for a comparison.

Timestep Selection. Depending on the application, it is often not necessary to have access to visitation maps for every possible time step. To exploit this, the temporal resolution of available visitation maps can be restricted. Defining a frequency F at which visitation maps should be available, events don't need to be stored for every time step but for every frequency cycle. All events occurring between two cycles are summed up in one entry, reducing storage requirements since only every F -th entry is stored. This reduction approach does not affect accuracy of visitation maps at available time steps.

Streamline Length. Shorter streamline length at creation time of the Visitation Graph results in a lower number of events and thus reduces the storage requirements of the Visitation Graph. While it is still possible to generate longer visitation maps from the resulting Visitation Graph, the accuracy is reduced after every integration time of T' .

Experiments on data reduction using Visitation Graphs as well as experiments on storage requirements and their dependency on the number of ensemble members can be found in Section 5.3 on page 88. An application of data reduction can be found in Section 5.4 on page 91.

5.2.4 Application to Visualization

As outcome of the previous results, we are able to effectively generate visitation maps from arbitrary start points with arbitrary initial distributions based on Visitation Graphs. This heavily contrasts with traditional visitation maps generated in ad hoc manner from integrated streamlines. They trigger a complete re-computation every time the start point is changed and considering multiple start cells with a given distribution results in vast computational effort. In addition, traditional streamline computation is always based on the whole ensemble, making this visualization approach unfeasible for ensembles with too many members to store. In our approach, to interactively create visitation maps from arbitrary start points, only the Visitation Graph needs to be stored. There is no need to store single ensemble members. Providing a tradeoff between storage requirement and accuracy, the Visitation Graph renders the exploration of ensembles possible that are prohibitive to store due to their size. Provided a Visitation Graph, different modes for visitation map creation are available.

Standard Mode. A single start point is selected. The visitation map of the desired length is calculated or approximated depending on T' .

Brush Mode. Multiple start points following an initial distribution are given. The visitation map is calculated for all start points at once. Different kernel functions are available as possible start distributions, they are scaled such that the integral over all initial cells is one.

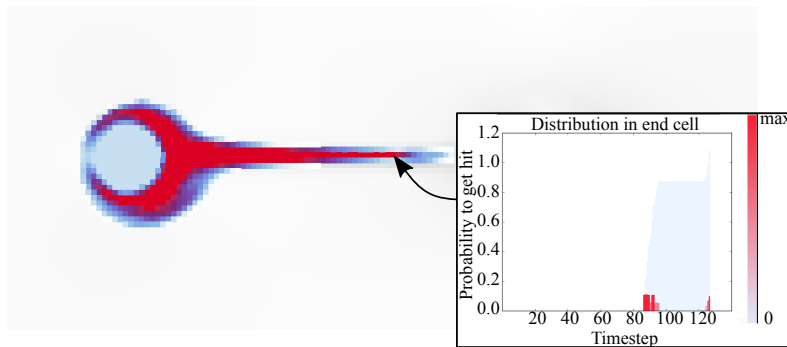


Fig. 5.9.: The **probabilities to be hit per time step** are shown after selecting a cell in a calculated visitation map: probabilities for individual time step and accumulated probability are shown in a separate window; this illustrates when heated fluid will reach the selected cell in the 2D convection ensemble (see Section 5.4 on page 91 for more examples).

Start-End Mode. Start points, a single end point and a probability p are defined. In each step of the visitation map calculation it is checked, if the given end point was reached with probability greater or equal to p . Calculation stops as soon as p or the maximal amount of time-steps is reached. For the end point the probability to be reached in every time-step is plotted.

When an end cell is selected, a plot showing the probabilities to be hit per time-step is shown in a separate window. It contains probabilities per time-step as well as the accumulated probability to be hit for the selected start distribution and end node (Figure 5.9). In addition, an animated, incremental visualization of the visitation map –time-step by time-step– emphasizes development over time in the given vector ensemble.

While for regular grids, the visualization of visitation maps (ad hoc or Visitation Graph based) is a straight forward mapping of scalars to colors, additional challenges arise when considering grids containing differing cell sizes or the same grid is evaluated in different resolutions.

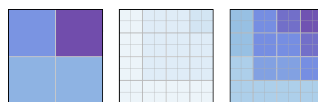


Fig. 5.10.: Appearance of the same area considering **different grid resolutions**: Left: coarse resolution, middle: no color compensation, right: color compensation. In every case the sum of the probabilities over all cells is one and the distribution is equal (Reproduced from [*117]).

The probability that a streamline passes by a grid cell depends on the cell's size. Since the sum of the probabilities over all cells in an area remains constant independent of

the resolution, small cells obtain smaller probability values in the visitation map than bigger cells. While from the mathematical point of view, this result is correct and intuitive, the resulting visualization might be misleading. Areas with smaller cells might appear less likely to be visited than coarser areas with the same probability. To handle this effect when plotting the visitation map, a compensation for the cell size can be used. The probability value assigned to each cell is multiplied by the size of the largest cell in the grid divided by the size of considered cell. This yields an intuitive visualization, see Figure 5.10 on the preceding page. However the plotted values no longer represent the actual probabilities, so both visualization options are accessible in our implementation.

To give an orientation in the explored vector field ensemble, visitation maps can be combined with any suitable static visualization in the background. The suitability of different visualization approaches has been reviewed by Iannis Albert under my supervision in his Master Thesis [*4]. He identified streamline approaches with selection strategies and clustering algorithms as the most promising approaches. However, in the case of not having stored single ensemble members, common techniques like streamlines can hardly be applied. To be still able to give an impression of the underlying field, we developed visualization techniques based on the Visitation Graph, intended to support the visitation map visualization:

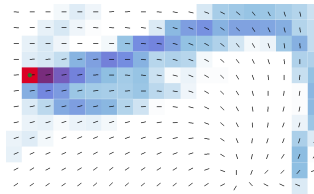


Fig. 5.11.: **Glyph visualization** indicates the average direction of connected cells in the Visitation Graph. Reproduced from [*117].

Glyph Visualization. Glyphs pointing towards the average direction of connected cells in the Flow Connection Matrix weighted with their probability can be drawn. That is, for every nonzero entry (c_i, c_j) in the Flow Connection Matrix, the midpoint of cell c_j contributes with weight $FC(t)(c_i, c_j)$ to the direction the glyph in cell c_i will point to. See Figure 5.11 for an example.

Degree Visualization. In- and out- degree of each cell in the Visitation Graph is plotted using different colors. Like this, areas with large variance and areas that are passed by many different streamlines are visible, giving an orientation of potentially interesting areas. See Figure 5.18 on page 95 for an example.

While exploration of vector field ensembles is the main purpose of our approach, it can be employed to visualize random fields as well: given a probability space (Ω, \mathcal{F}, P) , a random field of dimension $d \in \mathbb{N}$ is a family of random variables $\{Y(x, \cdot)\}_{x \in \mathbb{R}^d}$. A trajectory in a random field can be defined analogously to the certain case by an ordinary differential equation with a random field on the right hand side, which makes the equation a random ordinary differential equation (RODE).

$$\frac{dX_i(t, \omega)}{dt} = Y_i(X(t, \omega), t, \omega) \quad \text{with } t \geq t_0, \omega \in \Omega, i = 1 \dots d.$$

RODEs can be solved using standard numerical approaches such as Runge-Kutta methods or multi-step methods. Thus, instead of integrating streamlines in every ensemble member, multiple streamlines are integrated using samples of the underlying random field.

5.3 Experiments

To evaluate the behavior of the system, we conducted different tests on a standard workstation PC and an implementation in Python. Unless otherwise noted, a 50×50 testing ensemble containing 50 members created from normal distributions with constant $\mu = (1, 1)$ and varying correlation between -0.5 and 0.5 was considered for all tests. The cell size was uniform and the integration time T' was chosen as 60, while visitation maps of length $T = 100$ were calculated. All runtimes are measured on a purely serial execution of the code.

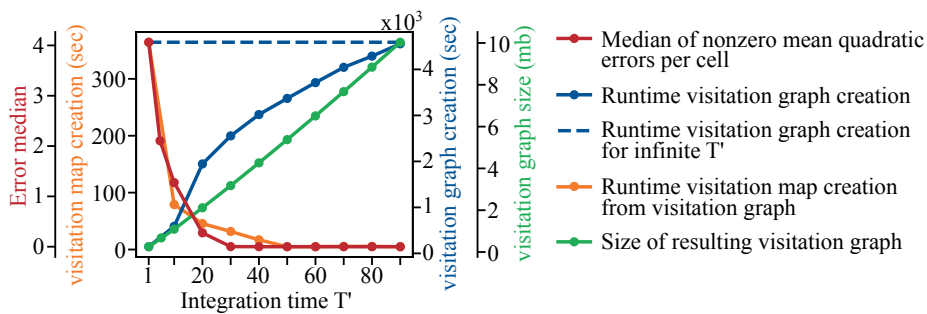


Fig. 5.12.: Visitation Graph tradeoffs: With increasing integration time T' , accuracy, storage requirements, and pre-processing time increase while the required time to create visitation maps decreases. The median of nonzero errors is shown since low errors are far more common than high errors.

Figure 5.12 illustrates the tradeoffs resulting in the generation of the Visitation Graph: a smaller number of streamline steps in pre-processing results in lower

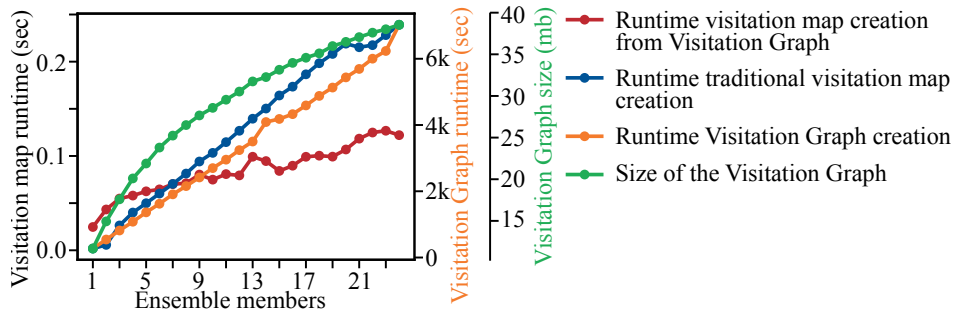


Fig. 5.13.: Dependency on member count: the more members are considered, the more beneficial is our method. The difference in runtime increases and already for very few ensemble members it outperforms the traditional ad hoc creation. Storage requirements for Visitation Graphs converge towards a fix maximal size. Visitation Graph creation grows linearly with the number of processed ensembles.

accuracy of the resulting visitation maps while time for Visitation Graph creation and storage requirements drop. Conversely, the longer the pre-processing integration time, the more accurate is the result while pre-processing runtime and storage requirements increase. The runtime for visitation map generation depends on the integration time T' : With fewer time steps stored in the graph, more calculations have to be executed to obtain the visitation map approximation (cf. Section 5.2.2 on page 79). Thus, time needed for visitation map creation decreases with increasing accuracy. As ground truth, we considered the Visitation Graph with infinite T' and corresponding visitation maps; every generated streamline is pursued until it either ends in a critical point, becomes local, or leaves the grid. For less streamline steps, the ground truth Visitation Graph is pruned at the desired step to avoid differences resulting from randomized streamline seeding. The mean squared error is calculated per cell as the difference between traditional visitation map starting in this cell and the visitation map generated from the Visitation Graph. As the number of pre-processing steps reaches the number of steps taken in the visitation map, the error drops to zero. No assembly is required and the exact visitation map is calculated.

In addition to the advantage of in situ pre-processing obviating the need to store every ensemble member to be able to calculate visitation maps, Figure 5.13 illustrates that already for a small number of ensemble members, the visitation map creation from the Visitation Graph is faster than the ad hoc creation. The larger the ensemble, the more beneficial is our method. This observation is strengthened when considering multiple start cells (cf. Figure 5.14 on the next page). To obtain accurate results for traditional visitation maps with multiple start cells under a delocalized start distribution, even more realizations are necessary to achieve adequate sampling.

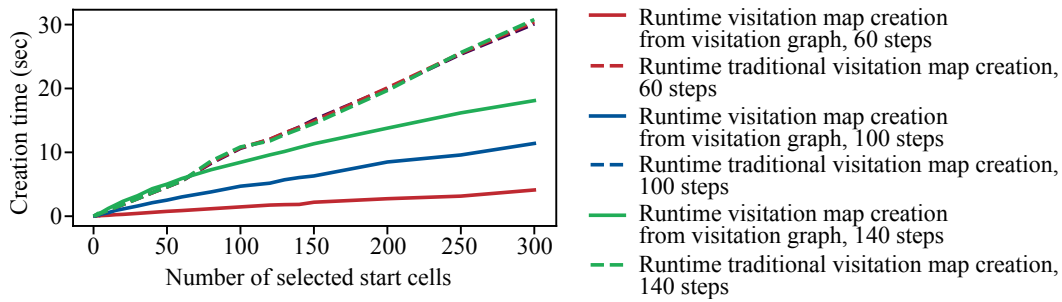


Fig. 5.14.: The **creation time of visitation maps starting in multiple cells**: visitation maps are faster generated from Visitation Graphs. The difference becomes larger with increasing number of start cells. In pre-processing, streamlines of 60 steps were calculated. Thus to calculate visitation maps with 100 and 140 steps, two resp. three calculation steps are needed. Intervals requiring the same number of steps result in similar generation times.

To examine Visitation Graphs as an alternative to downsampling, we executed the following experiment: our approach using coarse Visitation Graphs to obtain visitation maps was compared with the traditional approach to first sample down the data, then store it and create visitation maps in the traditional way. All approaches result in visitation maps of the same output resolution. A real-world and an analytical example were tested. The resolution was reduced from 168×168 to 80×80 and 336×168 to 160×80 respectively. We compared:

Our Method. A Visitation Graph of output resolution is calculated on high resolution input data. Using the Visitation Graph visitation maps are created.

Traditional. High resolution input data is downsampled to output resolution. Visitation maps are calculated in the traditional way.

Ground Truth (Test on Analytical Data). Exact streamlines in the analytical field are used to create exact visitation maps of the given output resolution.

Ground Truth (Test on Real-World Data). Visitation maps are calculated in the traditional way on high resolution input data, the final visitation map is sampled in the given output resolution.

Figure 5.15 on the facing page holds the results for the real-world example. Because of the exact ground truth, errors of our method in the analytical example were higher. While they reside in the same range as the errors of the traditional approach, the

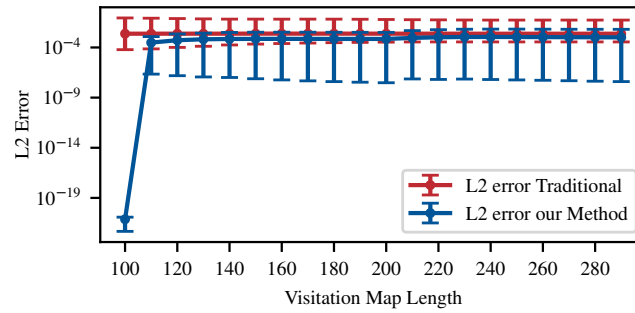


Fig. 5.15.: Visitation Graphs as an alternative to downsampling: Visitation Graphs are an optimal way to reduce data resolution and generate visitation maps afterwards: here, the integration time is set to $T' = 100$. Thus, for a visitation map length of 100, our method equals the ground truth. Also for higher $T > T'$, our method outperforms the traditional approach. The increase in the error of our method at a visitation map length of 220 is to be expected every time T is increased by T' . The error of the traditional method is higher and varies more.

maximal deviation only exceeds the one of the traditional approach after $T = 160$. For $T = 100$ again the error is much smaller than the one generated by traditional approximation. For $T \in [110, 160]$ our method gives a similar mean deviation and a smaller maximal deviation. So again our method outperforms the traditional approach not even for $T = T'$ but also for additionally approximated visitation maps for $T > T'$ up to a limit.

5.4 Results

While our contribution is a new calculation method for visitation maps, using visitation maps for visualization is well-established. Hence, this section aims more for giving details on applications of our method than on illustrating usefulness of visitation maps in general. We used the following data sets:

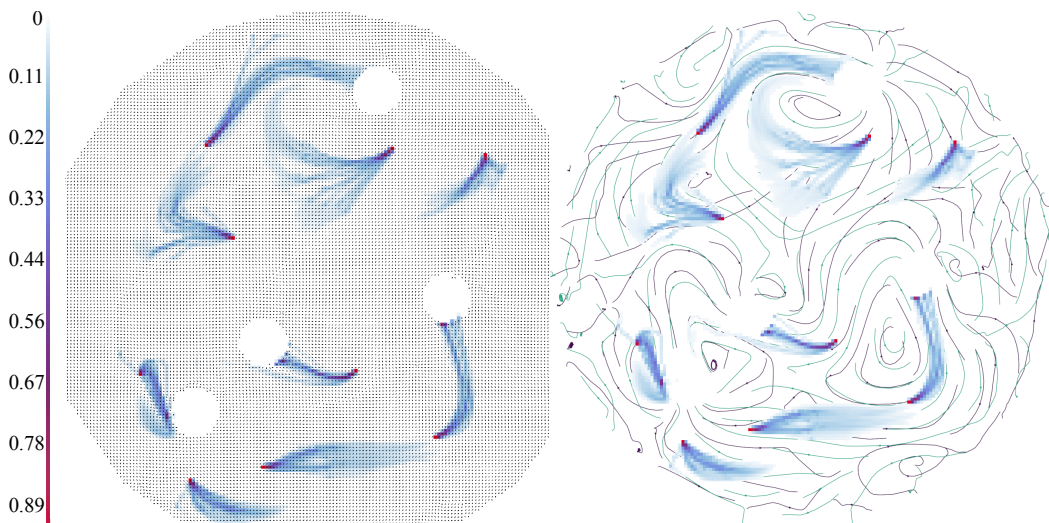
Industrial Stirring. Mixing in a stirring apparatus was simulated for 20 slightly differing viscosities resulting in a 2D flow field ensemble of size 168×168 . The device consists of two counter-rotating pairs of mixing rods that stir a medium in a cylindrical tank.

Convection Simulation. Simulating the flow around a hot pole as described in Section 3.4.3 on page 43. The 2D flow field ensemble is of size 128×256 .

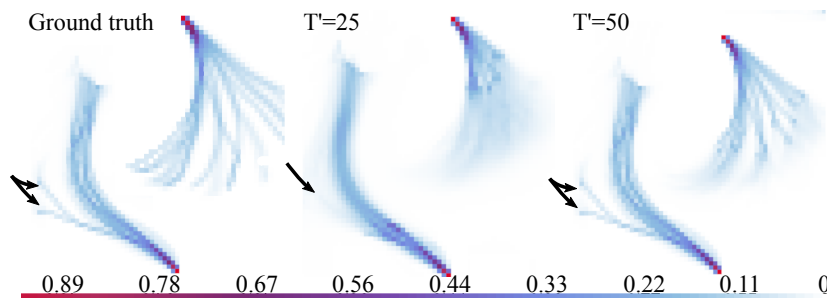
Cavity Flow. Laminar, in-compressible flow in a two-dimensional square domain where one border is moving with 1 m s^{-1} was simulated. The Reynolds number of the simulated liquid is increased by one between each member generation, resulting in 1990 ensemble members with Reynolds numbers between 10 and 2000. The resolution is 1000×1000 .

In all ensembles, a fixed time point was investigated using visitation maps based on a Visitation Graph.

5.4.1 Industrial Stirring Simulation



(a) Overview of the stirring simulation using multiple start points and different background visualizations: glyphs based on the Visitation Graph and streamlines of two ensemble members. $T' = 50$, $T = 100$.



(b) Results for $T' = 25$ (center) and $T' = 50$ (right) are compared with the ground truth (left). The first approximation after 25 and 50 steps respectively is visible.

Fig. 5.16.: Industrial Stirring: overview in (a) and detail for different integration lengths T' in (b).

This example illustrates the approximation of visitation maps using Visitation Graphs based on shorter streamlines and different choices for T' . Two pre-processing

integration times were used. Serial runtime for $T' = 50$ with about 40 minutes per member is significantly higher than runtime for $T' = 25$ (about 20 minutes). The number of realizations was estimated for the latter execution. Based on a bunch size of 10, cells with very small vectors, especially the ones where no data exists, were start point for much less streamlines than in the non-automated calculation with 100 realizations for every cell. An exploratory visualization of the stirring simulation is shown in Figure 5.16a on the facing page. Combining multiple start points with glyph visualization or streamline visualization of multiple members gives an overall impression. Additional initial points can be selected interactively. Like this, interesting regions, for example the flow behavior around the mixing rods can be examined easily by starting visitation maps close to them. The resulting visitation maps provide the opportunity to **compare** the behavior of the ensemble members and **separate** possible outliers, for example the marked members in Figure 5.16b on the preceding page. With a pre-processing streamline length of 50, visitation maps of length 100 were generated.

In Figure 5.16b on the facing page, visitation maps for $T' = 25$ and $T' = 50$ are compared. The error induced after the first approximation step is clear to see. Still, the tradeoff between pre-processing time and accuracy is profitable for exploration. The early approximated visitation maps are as useful as the ones based on $T' = 50$. In addition, the storage requirement of the Visitation Graph drops from 380MB to 118MB. Compared to the original ensemble size is 4.5MB, this is an increase. Yet, this example does not aim on reduction of the data size. With 20 members of size 168×168 , the number of ensemble members is not large compared to the number of cells. The average numbers of edges are 163.4 and 72.0 for $T' = 25, 50$ respectively, confirming that in real world examples the upper bound of $m \times n = 28,224$ is vastly overestimating the number of outgoing edges. The average number of stored 4-tuples per edge is 10.3 and 7.3 respectively which is again much smaller than the upper bound of T' in both cases.

5.4.2 Convection Simulation

Using the convection simulation, we illustrate interactive exploration possibilities using visitation maps that are based on Visitation Graphs. A Visitation Graph with $T' = 100$ was generated for a single time step of the convection simulation (Figure 5.17 on the next page). First, the area of turbulent behavior is detected by seeding around the pole and **combining** the resulting visitation maps (5.17a). This area can then be explored by interactively adding multiple start points, allowing to **compare** and **separate** areas with different flow behavior. Figures 5.17b to 5.17d

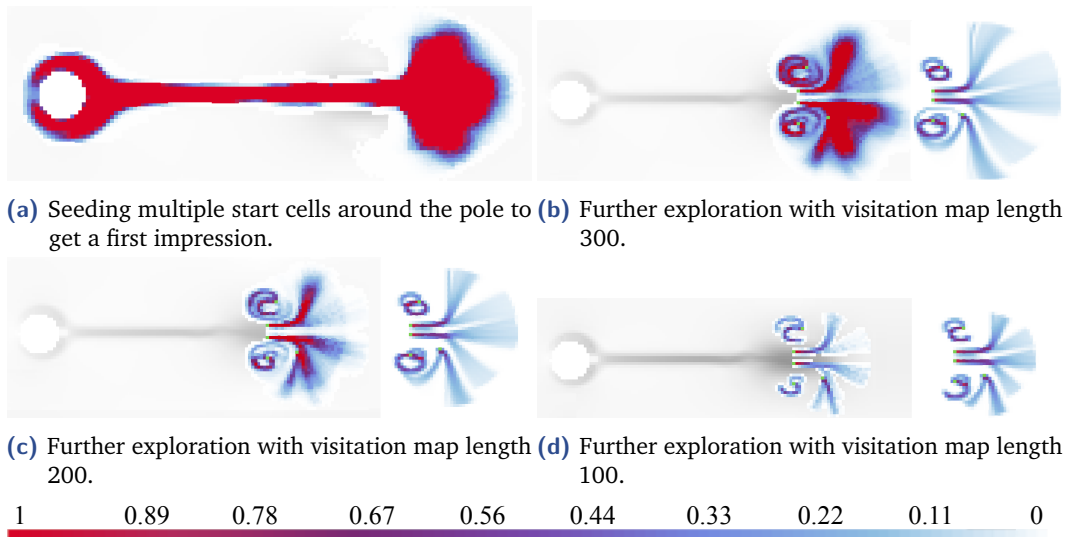


Fig. 5.17.: Convection simulation: A first impression in (a) and further exploration with decreasing visitation map lengths; for orientation the average velocity over all ensemble members is given in the background. Processing one member took 145 minutes. Again, the upper bound of $m \times n = 32,768$ outgoing edges per cell is much higher than the real number which is 18.8 on average. 14.8 time steps are stored per edge on average which is again much less than the upper bound $T' = 100$. This results in raw data of 72.8MB compared to original data size of 5.9MB. While our approach is not able to reduce the data in this case with a relatively low number of ensemble members, the speedup in visitation map creation is still given.

show this process using different visitation map lengths; for orientation the average velocity over all ensemble members is given in the background. Processing one member took 145 minutes. Again, the upper bound of $m \times n = 32,768$ outgoing edges per cell is much higher than the real number which is 18.8 on average. 14.8 time steps are stored per edge on average which is again much less than the upper bound $T' = 100$. This results in raw data of 72.8MB compared to original data size of 5.9MB. While our approach is not able to reduce the data in this case with a relatively low number of ensemble members, the speedup in visitation map creation is still given.

5.4.3 Cavity Flow

Using the cavity flow example, we illustrate the data reduction abilities of Visitation Graphs. On the cavity flow data set with a high number of members, a data-reducing Visitation Graph of resolution 100×100 was created with a time step-storing frequency of 10 and $T' = 20$. While creating visitation maps based on the original resolution might be impossible, the resulting Visitation Graph is able to retain information from the original resolution and still reduce the data. With 8 cores and a speedup factor close to 8, the calculation time of the Visitation Graph is about 140sec per member for $T' = 20$ and 100 streamlines per cell. Instead of the

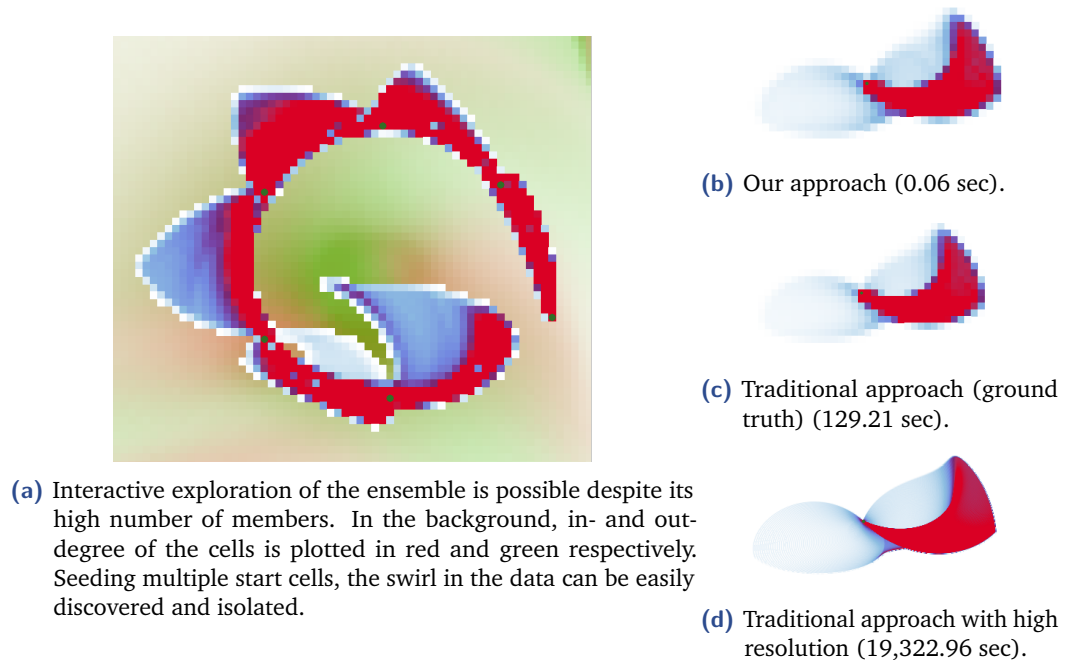


Fig. 5.18.: The **cavity flow simulation**: Interactive exploration example and approach comparison.

30.2GB large ensemble, the Visitation Graph with 43.7MB can be stored and used to create visitation maps. On average, 131.67 outgoing edges were stored per cell and on each edge on average 1.15 time steps (this number is reduced due to the chosen storing frequency). In Figure 5.18, an example for incremental exploration of the field with degree visualization background is given. Visitation maps generated from Visitation Graphs and in the traditional way are compared and a visitation map for the fine resolution is given.

5.5 Discussion and Conclusion

Vector field ensembles can be explored in an intuitive way using visitation maps. Traditional ad hoc calculation of visitation maps requires every ensemble member to be stored and sampled. Thus, it can not be applied to ensembles that have a large number of members. In addition, every change of the seed point (or initial distribution) requires a full re-computation of the visitation map. And an initial distribution that is not strongly localized, results in even more time expensive computation. Visitation Graphs provide solutions to these problems by isolating the sampling in in situ pre-processing while allowing the computation of visitation

maps in an interactive fashion, also for large ensembles and delocalized initial distributions. Costly calculation time is shifted to in situ pre-processing.

While the speed up of visitation map generation is present already for small ensembles, our method provides substantial advantages for ensembles with many members. For these ensembles, storage savings can be accomplished and visitation map generation speed up is quite significant. Multiple additional storage saving techniques are available for Visitation Graphs, allowing in situ data reduction that is superior to other approaches with respect to visitation map creation.

The Visitation Graph stores information about all events of all generated streamline samples. Depending on spatial resolution, storage requirements can still be significant. A possible aggregation of the data in addition to a set storing frequency is a topic for further research. Furthermore, parallelization (potentially on GPUs) can provide dramatically better runtimes for Visitation Graph generation. Further potential for runtime reduction lies in re-using of generated streamlines during Visitation Graph generation.

Representing vector field ensembles as their corresponding Visitation Graphs offers a plethora of opportunities for further research: graph algorithms and clustering could be applied (e.g. in similarity to [190]), to exploit all strengths of this representation. Considering multiple time-steps of time-dependent fields, results could be compared by comparing the Visitation Graphs, and missing time-steps could be interpolated on the graph level. Concerning visualization, different techniques could be applied to present the calculated visitation map such as isocontours or using transparency.

Finally, generalization of Visitation Graphs to three-dimensional random fields is another research direction. While the basic technique of Visitation Graphs can be transferred quite naturally to three dimensions, the increasing number of grid cells will be the main challenge. The data reduction techniques presented in this paper seem to be promising to deal with this challenge. In addition, a link between Visitation Graphs and graphs used for three dimensional space partitioning is an exciting research topic. Furthermore, the generalization of visitation maps to three dimensions requires an evaluation of existing three dimensional scalar field visualization techniques for this special purpose.

Part II

Uncertainty in Automated Data Analysis



Automated data analysis is a crucial approach when dealing with large data sets and/or insufficient time for analysis. With increasing amounts of generated data and decreasing expected reaction times, automated analysis is now a widely applied approach, supported and fueled by research progress in artificial intelligence. Examples span from tracking pixels that automatically analyze user behavior, to automated profiling and in situ analysis. In many cases, only the processed data is stored and further used. In some cases however, results of the automated analysis trigger reactions. For example automated monitoring of stock prices trigger purchase or disposition. In case of important decisions, these are not left to the system, but analysis results and induced reaction are verified by humans. This requires precise, easy and fast to interpret visualizations.

Visually Supported Anomaly Detection in Cyber Security

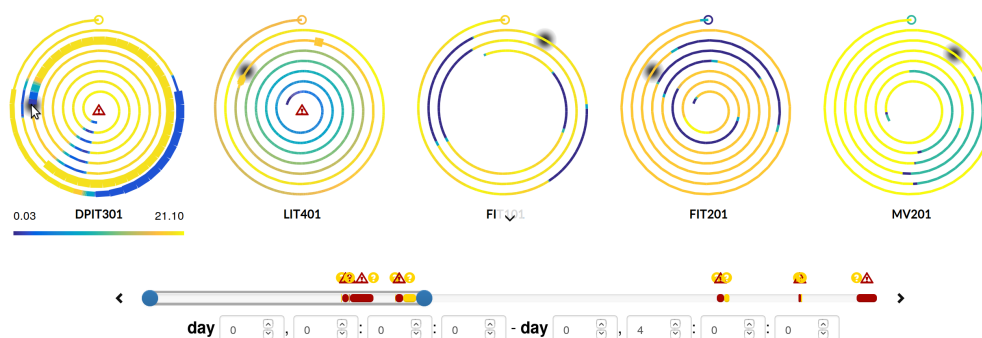


Fig. 6.1.: Overview of the **Security in Process System** supporting triage analysis in operational technology networks by presenting device readings and results from automated anomaly detection to the analyst.

In cyber security, reaction times are crucial. Scripts and tools lower the effort that is necessary to attack systems, while the amount of sensitive data that is stored online and the effort to secure distributed systems increases. Hence, automated anomaly detection is essential and even automated emergency responses are thinkable. Yet, current anomaly detection algorithms often lack certainty of their results, requiring the additional verification by analysts. To allow analysts to take part in the anomaly detection process, a visualization providing a quick overview but a sufficient amount of details about the current data is required. Developing time series visualizations for different purposes is an active field of research. A thorough summary on visualization of periodic and aperiodic time series data was given by Aigner et al. [3].

Cyber security not only has to be ensured in information technology networks, but also in *operational technology networks* (OT networks). In contrast to IT networks, OT networks consist of devices and infrastructure that monitor and control physical industrial processes. An example is a water treatment plant. All *devices* –sensors, pumps, and other actuators, that control for example the dosage of added chemicals– are part of the OT network, connected by programmable logic controllers (PLCs). Because of the underlying industrial processes, device readings in OT networks oftentimes develop periodic patterns; tanks are filled and run empty until they are filled again, the concentration of added chemicals decreases during the purification

process until a threshold is reached and additional chemicals are added again. Hence, missing periodicity represents an anomaly in OT networks. This is a strong contrast to IT networks, where periodic behavior is a signal for malware activity [88].

Examples for malware detection based on periodical behavior are the application of the discrete Fourier transform by Gove et al. [72] and the Fourier transform based periodicity detector for network traffic by Huynh et al. [5]. They use circular plots to compare periodic behavior of different alerts.

Over the last two decades, automation, and thus the use of OT networks in industry have increased rapidly, as have attacks on such networks [51]. Network environments that are difficult to update and the use of communication protocols that do not contain authentication or encryption lead to high vulnerability once an intruder has successfully breached the communication network [134, 135]. In addition, historical reasons caused these networks to be less secured against attacks than deemed appropriate for home and office IT [89]: first, OT and IT networks are supposed to be physically separated. Second, attacking OT networks is expected to be difficult due to their highly application specific implementation. However, both of these aspects have become obsolete nowadays: commercial off-the-shelf products in the industrial area (for example PLCs) facilitate set up, maintenance and operation of industrial applications by using common interfaces and programming libraries. This also decreases the difficulties for attackers. Furthermore, new use and business cases introduced as part of Industry 4.0 efforts break the physical separation of networks [187]. Relying on the communication and computation capabilities of (industrial) internet of things devices, access routes to OT networks are created. Even if no such access is possible, attackers have successfully managed to move laterally to the OT networks in the past after breaking the IT network perimeter [52].

During the Covid-19 pandemic starting in 2019, cyber attacks further increased due to new possibilities to attack (hard and software in home office, but also anxiety of people). Lallie et al. give a detailed timeline of attacks during the pandemic [108]. Especially the healthcare industry and its OT networks were attacked, capitalizing their vulnerability during the pandemic or aiming on stealing intellectual property such as data relating to COVID-19 vaccine development, and treatment options [141]. Hence, additional actions to secure OT networks are required, but outdated structures and difficulties to update the components are problematic. This has led to recent research in cyber security with the aim to detect attacks in already available information.

Industrial intrusion detection based on device data is a widely regarded topic in cyber security research. Anomalies in the data are detected with a plethora of different approaches. Schneider et al. use autoencoders to detect anomalies in cyber-physical system networks [168], Goh et al. and Feng et al. use neural networks for the detection [58, 71]. One class support vector machines are presented by Maglaras et al. as a machine learning algorithm to detect novel and unknown attacks [128].

Results from different anomaly detection approaches have been visualized in different ways: Stoffel et al. provide a visualization combining multiple data sources that monitor a computer network [185]. Their visualization relies on well-known time series visualizations. Combining these visualizations and highlighting of detected anomalies facilitate shape, correlation and pattern recognition. Karapistoli et al. detect selective forwarding and jamming attacks in wireless sensor networks [100]. Findings are then incorporated in graph visualization and crossed views, providing an efficient and fast overview of the network status. Boschetti et al. refrain from incorporating information on detected anomalies directly in visualization [23], they use plots, histograms, graphs and matrix visualization to provide information on network traffic. Timesteps with a potential attack can be selected from a list and are linked to the different visualizations.

Although many visualizations for anomaly and periodicity detection exist, to our knowledge none consider detection of anomalies in device data of industrial processes. Visualizations designed for anomaly detection in network activity are not suitable for this task since the typical periodic behavior is not taken into account or is even classified as an indicator for malware activity. In OT networks, the periodic nature can be exploited using spiral plots as proposed by Weber et al. [208]. A three-dimensional adaption of spiral plots was given by Tominski et al. [195]. Hu et al. use spiral plots to visualize alarm floods and their patterns in complex industrial facility monitoring [86].

Using this idea, I developed a spiral plot based time-series visualization in collaboration with Dr. Simon D. Duque Anton (DFKI Kaiserslautern), Heike Leitte (Visual Information Analysis Group) and Christoph Garth, published and best paper at VizSec 2019 [*116]. Besides our presentation at VizSec, we also presented our work in an invited key-note at BKA Praxisforum 2019, titled “Detektion und Ermittlung von Angriffen auf OT-Netzwerke und IoT-Netze” (“detection and determination of attacks on OT-networks and IoT networks”). Together with Dr. Simon D. Duque Anton, I developed an anomaly detection approach [*6, *7]. Its results provided the basis for my work with Prof.Dr. Heike Leitte and Prof.Dr. Christoph Garth, resulting

in a supporting visualization system for triage analysis in OT networks – the *Security in Process System*.

In this System, the spiral plots combine device data with the anomaly detection results. Like this, it provides a simultaneous overview of measurements and anomaly detection results. The intuitive visualization allows the supervision of the system behavior in normal operation and fast detection of attacks not only for cyber security trained personnel, but also for laymen. Hence, an improvement of security by supervision of the data is also possible in factories without dedicated cyber security personnel. The visualization was designed and evaluated to support triage analysis; it is presented in Section 6.3. Possible analysis strategies that can be used to verify attacks, analyze their causes, and to detect false alarms are described in Section 6.4; an expert evaluation is provided in Section 6.5 and user studies for our system and a further developed version incorporating knowledge assistance are given in Chapter 9.

Again, the general visualization tasks can be addressed with the system: using the Security in Process System, it is possible to **compare** time series and their development, identify and **combine** similar behavior within the readings of a single device (f.ex. detecting periodicity) and between devices, and **separate** intervals with abnormal behavior from normal behavior.

6.1 Application Background and Challenges

In our application, the data sources used to detect anomalies are PLCs monitoring a modern six-stage process of water treatment. The data set contains a real-world underlying process with introduced attacks and is provided by the iTrust, Centre for Research in cyber security, Singapore University of Technology and Design [91, 129]. All attacks are documented, so a ground truth comparison of the results is possible.

To provide an overview of our visualization system and its strengths, I track the development of two components with illustrative results, namely sensor DPIT301 and actuator LIT401. For purpose and location of these components in the considered water treatment process see Section 6.4 on page 113.

6.1.1 Automated Anomaly Detection

The setting of monitored device data in industrial processes differs fundamentally from monitored network activity in IT networks. While in the latter, periodic patterns indicate attacks, in the former the absence of frequent patterns or periodicity indicates anomalies. Thus, three different approaches for anomaly detection based on prediction and comparison of the measurements were implemented and evaluated by Duque Anton et al. [*6], namely one-class support vector machines, isolation forests and matrix profiles. I contributed in development and implementation of matrix profile approaches. While one-class support vector machines and isolation forests are one-class classifiers that analyze the data on a packet basis, i.e. step by step, matrix profiles are used to perform a time series analysis of a given complete time frame. One-class classifiers require extensive training with a large amount of data. This is time consuming and the data might not be easy to acquire. Even though the detection capabilities of matrix profiles are increased with a larger data base for comparison, they do not require formal training. Furthermore, compared to tuning on hyper-parameters of the used one-class classifiers, matrix profiles require very little tuning. This makes the detection easy to set up and robust to different kinds of data. Based on these facts and the obtained results for the considered data set, matrix profiles turned out to be most suitable for the task of anomaly detection. The results shown in this part all base on anomaly detection using this approach.

Matrix profiles were developed by Yeh et al. as an algorithm for motif discovery [217]. An interactive application can be found on their web page [130]. Matrix profiles are based on the comparison of intervals of the analyzed data with the remainder of the data. A distance (e.g. z-normalized distance) between the sub-series of a given length m and all other sub-series of the same length is calculated in a sliding window fashion. The u-normalized distance is calculated as follows:

$$d(x, y) = \sqrt{\sum_{i=1}^m (\bar{x}_i - \bar{y}_i)^2} \quad \text{with} \quad \bar{x}_i = \frac{x_i - \mu_x}{\sigma_x}, \quad \bar{y}_i = \frac{y_i - \mu_y}{\sigma_y}$$

Where σ_x and μ_x are standard deviation and mean value respectively.

$$\mu_x = \frac{\sum_{i=1}^m x_i}{m}, \quad \mu_y = \frac{\sum_{i=1}^m y_i}{m}$$
$$\sigma_x^2 = \frac{\sum_{i=1}^m x_i^2}{m} - \mu_x^2, \quad \sigma_y^2 = \frac{\sum_{i=1}^m y_i^2}{m} - \mu_y^2.$$

After applying Pearson's Correlation Coefficient as described by Benesty et al. [17]

$$\text{corr}(x, y) = \frac{E((x - \mu_x)(y - \mu_y))}{\sigma_x \sigma_y} = \frac{\sum_{i=1}^m x_i y_i - m \mu_x \mu_y}{m \sigma_x \sigma_y},$$

the Euclidean distance relates as described by Mueen et al. [139],

$$d(x, y) = \sqrt{2m(1 - \text{corr}(x, y))}.$$

The resulting metric for distance calculation is then given by

$$d(x, y) = \sqrt{2m \left(1 - \frac{\sum_{i=1}^m x_i y_i - m \mu_x \mu_y}{m \sigma_x \sigma_y} \right)}.$$

The minimal found distance is then used as measurement for the anomaly of the current interval, the anomaly score. Thus, the algorithm checks if the considered development re-occurs at other time points. To avoid missing attacks that are executed multiple times, we implemented a counter for similar intervals and took it into account when rating the anomaly probability. If an event is found more infrequent than other events, it is likely to be an anomaly.

The final result of the anomaly detection algorithm is a score for every time step and every device that indicates the probability of an anomaly in the readings. These scores are divided into three categories, based on the height of the anomaly rating:

- I values where an anomaly is unlikely,
- II values that are extraordinarily high but are not high enough to clearly indicate malicious activity and
- III values where an anomaly is very likely.

The thresholds used for this categorization can be adapted to the used anomaly detection approach. If not meaningful in a given context, category II can be omitted from analysis by setting the thresholds accordingly. The thresholds in the presented application have been chosen based on the values obtained from anomaly detection on normal data. Having a small range in which values are classified as category II directly above the range obtained when considering normal data provides a buffer for possible normal values with slightly higher results and thus increases the credibility of category III. While abnormal behavior might arise from sources different than malicious activity, it should always be noticed and investigated. Furthermore, the basic idea of using anomaly detection to detect attacks is that attacking the system will lead to abnormal values.

6.1.2 State-of-the-art analysis

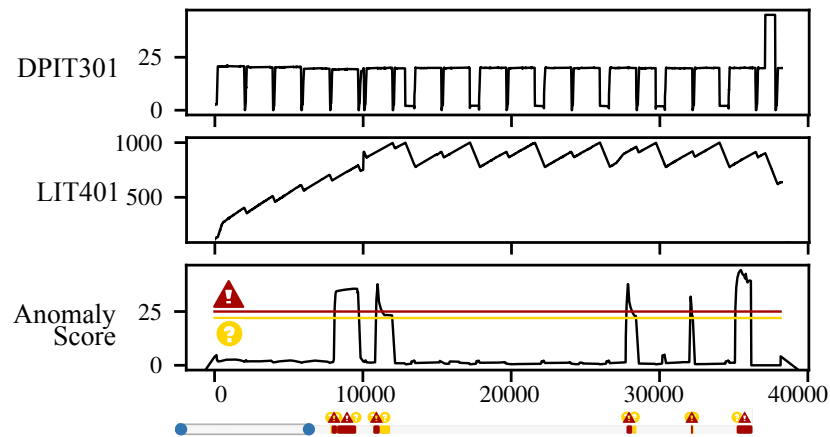


Fig. 6.2.: Anomaly detection results: (top to bottom) readings of two devices, anomaly detection results for DPIT301 with chosen thresholds for categories II and III, and the corresponding time slider.

Up to now, verification of detected anomalies was performed using basic time series visualizations as in Figure 6.2. Five anomalies are detected in the given time frame using matrix profiles. While the matrix profiles detect anomalies with a sufficiently high probability, they do not indicate why an anomaly was detected. Identifying the reason for the anomaly detection is challenging in this setting, as well as the assessment of detected anomalies as true or false positives. For example, moderate changes in the periodical behavior of the data are missed easily. Hence, a more elaborate visualization is needed to support triage analysis, making use of the periodic behavior of devices and supporting the understanding of anomaly detection results.

6.2 System Requirements

While anomaly detection is a first step towards an early and fast detection of attacks on the industrial process, its results need to be visualized for verification. The following requirements for our visualization system have been identified in cooperation with cyber security experts:

- R1 System monitoring and triage analysis should be supported simultaneously.
- R2 Detected anomalies should be clearly highlighted in the data.
- R3 Classification of values in category II as abnormal or normal, and
- R4 identification of false positives should be possible.

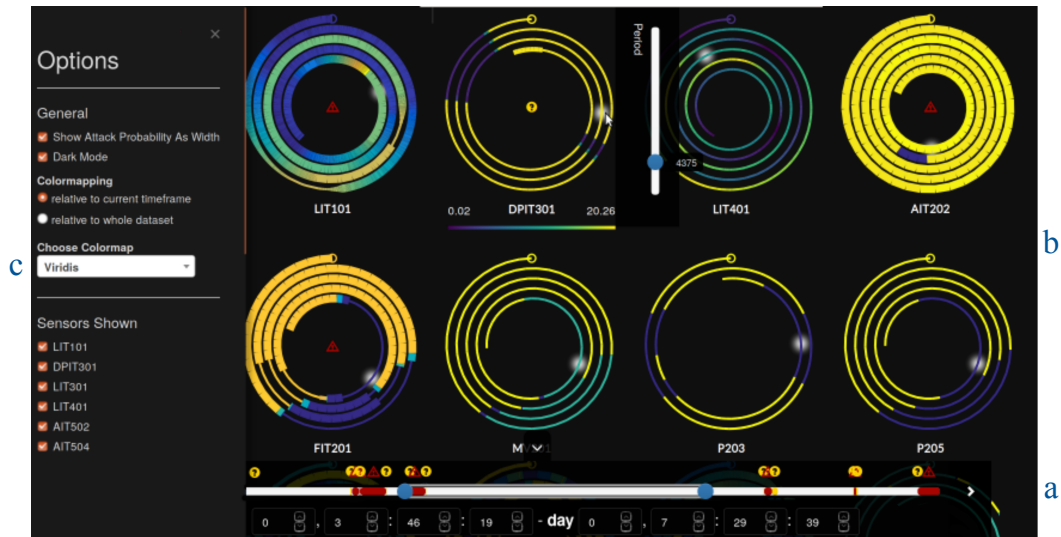


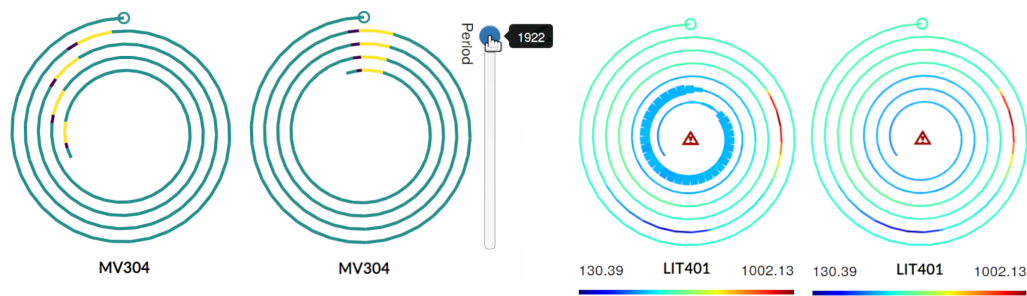
Fig. 6.3.: The Security in Process GUI: **a** Time slider, **b** spiral chart and **c** options menu. The time slider gives an overview on the complete available data and contained anomalies. In the spiral chart, every device is represented by a spiral plot winding from the center to the border. It encodes the measured data in color and the anomaly score as line thickness.

- R5 The displayed information and the interaction possibilities should allow identification of false negatives.
- R6 The visualization system should render triage analysis by experts as well as by laymen (in terms of cyber security) possible.

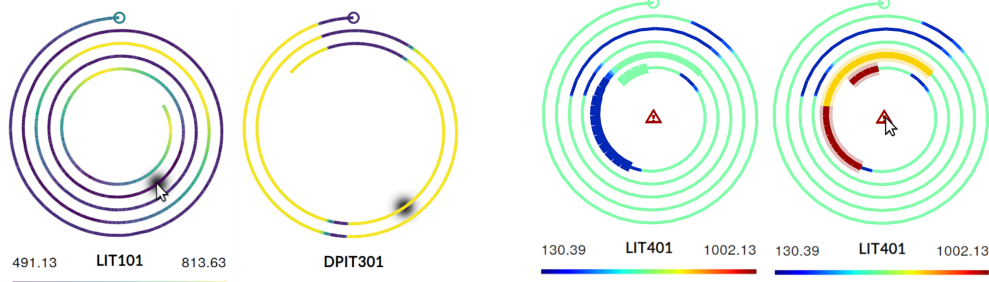
Depending on the algorithm used for anomaly detection, data for live monitoring of the system is available: whiel Matrix Profiles only allow analysis of a fixed time frame, the other algorithms work on packet basis and can be applied in real time. Hence, live monitoring of the system should be possible where new values are added to the data set at a given frequency.

6.3 The Security in Process System

We designed the Security in Process System according to our defined system requirements.



(a) **Period adjustment:** automatically detected (left) and manually adjusted period (right). (b) **Anomaly score visualization** as line thickness can be switched on (left) or off (right).



(c) **Spotlights:** Highlighting the hovered time step in every spiral simplifies linking of concurrent features. (d) **Anomaly highlighting:** Hovering the alert sign in the center of a spiral (left), areas of categories II and III are colored accordingly and line thickness is animated (right).

Fig. 6.4.: Spiral plot features: period adjustment, anomaly score visualization, spotlights and anomaly highlighting.

6.3.1 User Interface

The user interface consists of three main parts that are shown in Figure 6.3 on the preceding page. The *time slider* (a) represents the complete data set and provides an overview and the temporal navigation for the *spiral chart* (b). This chart represents the main area of the visualization system. It provides a detailed view on the time interval chosen in the time slider using one spiral plot per selected device. Figures 6.4a to 6.4d illustrate the introduction of this component's features. The *options menu* (c) is hidden by default. Here, all visualization options can be chosen and devices can be included or excluded from visualization. Also, the theme of the whole visualization system can be switched between light and dark mode. Requirements leading to decisions in the design process of single features are named in parentheses.

6.3.2 Overview and Detail

As shown in Figure 6.2 on page 107, the time slider represents the entire data set and highlights detected intervals of categories II as warning ⚠️ and III as alert 🚨.

This is done across all available devices, providing an overview of the process status (■R1, ■R2). From the complete available data, a time frame can be selected interactively to obtain detailed information in the spiral chart. The maximum size of the selected time frame is set to four hours, covering 14,400 values at one data point per second. This size was chosen based on the experience of the involved security experts. The size of the spiral plots was chosen accordingly. If needed, spiral plot size and the maximum visualized length can be adapted easily.

The spiral chart b in Figure 6.3 on page 108 visually encodes device data and anomaly detection for the selected time frame. To fully exploit the periodicity of the data and the possibility to detect anomalies visually, we chose spiral plots for this visualization (■R1, ■R2, ■R3-6). Our configuration of spiral plots is the following:

The time axis is winding from the center of the spiral to the circumference. That is, the most recent time point is always shown in the largest, outermost circle at the top, highlighted by a small circular marker in corresponding color (■R1). This is especially helpful when using the system in live mode, where new data is streamed continuously in the visualization. New data points are added on the largest, outermost circle of the spiral, providing most details for the most recent time interval. Changes in the streamed measurements are accentuated by the change of the marker's color. As the endpoint of the spiral is fixed, widening the visualized time frame results in growing of the spiral towards the center. The earliest visualized time point is represented by the end of the spiral closest to its center.

The initial cycle length of each spiral is estimated using the zero-crossing method on the corresponding device readings. As suggested by Weber et al. [208], the cycle length of each spiral can be manipulated interactively, thus taking advantage of the user's pattern recognition abilities to **combine** similar patterns and increase the accuracy of the presented period length (see Figure 6.4a on the previous page). It also allows an adaption of the cycle length to a change in the data's periodical behavior during analysis (■R1, ■R2, ■R3-6). Such a change could for example be provoked by operator interventions as explained by Caselli et al. [35]. An example is given in Section 6.4 on page 113. Increasing the cycle length, more time steps are visualized per cycle and the overall spiral becomes shorter, with less revolutions. Decreasing the cycle length results in fewer time steps per cycle and thus in a

longer overall spiral with more revolutions that ends closer to the center. Since the periodical behaviors of different device readings are a priori independent, the period for each spiral plot is treated individually.

Colors in the spiral represent the different device readings either relative to all available measurements obtained by the considered device or relative to the currently visualized time frame. The first option is suitable to **compare** values of the currently selected time frame to the overall values while the second one gives more detail on behavior of the data in the current time frame. Among others, the Parula colormap is available which is suitable for people with color blindness.

The pre-attentive attribute of line thickness [9] is used to provide information on the calculated anomaly score at each time point, i.e. the probability for an anomaly calculated by the employed anomaly detection algorithm (■R2, ■R6): the thicker a line is, the more likely an anomaly occurred. Values of category III are of a given maximal thickness, values of category I of a given minimal thickness and values in between (category II) interpolate between maximal and minimal thickness. To focus on device readings and be able to **compare** them, the spiral plot can optionally be drawn with a constant line thickness (■R1) as shown in Figure 6.4b on page 109.

As soon as the cursor hovers over any spiral on screen, the corresponding colormap and value range are shown, and a spotlight follows the cursor to facilitate linking between time points on spirals, their **comparison** and the **combination** of similar patterns across devices. This spotlight will always highlight the point on the spiral that is closest to the cursor. Simultaneously, spotlights in all other spirals highlight the corresponding time point according to their cycle length in linked views. Due to the fact that every spiral shows the same number of time steps but with different cycle length, outermost and innermost points of all spirals coincide. In between, the speeds of the spotlights differ. Keeping the spotlight behind the actual spiral visualization enables the user to easily **compare** single data points at each visualized time point (■R1, ■R3-6) (see Figure 6.4c on page 109).

6.3.3 Anomaly Highlighting

Detected (potential) anomalies are highlighted in the time slider as well as in the spiral chart (■R2) to **separate** them from normal readings. Across the entire visualization system, categories II and III are marked consistently with colored areas and symbols: ☺ and ▲. If both categories II and III are present, the highlighting for the more problematic category III is chosen.

In the time slider, time frames with suspicious readings in any device are highlighted. The slider bar is colored accordingly in these areas. Centered above the colored area, the corresponding symbol is shown. Clicking on this symbol, the selected time frame is fitted to the corresponding area. In addition, all devices where the (potential) anomaly was detected are selected for visualization, ensuring that the users are able to identify all affected devices. After a time frame including data points from categories II or III was selected, further highlighting is provided in the spiral chart.

In this detailed view, to emphasize the occurrence of (potential) anomalies in the data, the corresponding symbol is displayed in the center of spirals that contain data categorized as II or III. See for example devices LIT101 and DPIT301 in Figure 6.3 on page 108. Especially if the anomaly score visualization is deactivated and the line thickness of the spiral plots is constant, these symbols ensure that attacks are still clearly visible. Hovering over the symbol in a spiral's center, data points contained in the spiral plot of categories II and III are highlighted in the corresponding colors. To further draw attention to the highlighted areas and to ensure that the highlighting colors are not mistaken as part of the used colormap, line thickness is animated (see Figure 6.4d on page 109) (■R3, ■R4, ■R6).

6.3.4 Interaction

Multiple interaction opportunities are provided to support monitoring of the process and triage analysis of detected anomalies (■R1, ■R6):

Navigating the overall data by manipulating the time slider is possible in multiple ways: The borders of the selected time frame can be adjusted by dragging the border markers or clicking on the slider. If the maximum size of the selection is exceeded, the border that is currently not dragged is adjusted to obtain a valid interval. To shift the entire selected time frame while keeping its current size, the arrow buttons next to the slider can be clicked. If available, precisely known time points can be entered directly below the slider in the time frame information. Clicking on symbols indicating potential anomalies adjusts the time frame to include all corresponding time points and selects all affected devices for visualization.

After the time frame is fixed, the updated plots on the spiral chart can be adjusted individually. Clicking on a spiral, a slider to interactively manipulate the spirals cycle length is shown. Options applying to all shown spiral plots are available in the options menu. The colormap and its borders can be changed and visualizing results from anomaly detection as line width can be toggled. To be able to focus on individual device readings, the visualization can be limited to a subset of devices.

6.3.5 Implementation and Scalability

The visualization system was implemented using web technologies (HTML, CSS, and JavaScript) in the interest of portability and maintainability. The controls are chosen to permit usage of the visualization system on touch screens. Significant speedups compared to a naïve implementation were gained by rendering the spirals using a given threshold ϵ and a maximal line length k . A line is started at the first drawn time step and traced step by step with step size defined by the spiral's cycle length. As long as the first encountered data point of the line does not differ more than ϵ from the following data points, a single line with a single color is drawn. The line either ends when a higher difference occurs or the maximal length k is reached. Hence, there are less lines drawn than time steps are contained in the spiral, leading to a speedup in rendering time. This precludes a highlighting of single time steps in the spiral, therefore the spotlight calculation does not rely on drawn elements but is purely analytical. Spirals were implemented using the HTML SVG element, re-calculation and re-drawing were minimized. In our tests, 51 different devices were shown at a time and smooth real time-interaction was possible. This holds also true when settings are changed.

Showing significantly more than the 51 devices would result in longer loading times, and scrolling a long page full of devices would not be optimal. Having different sub-pages with a meaningful division of the devices would facilitate the navigation, provide a better overview and allow the system to asynchronously load additional device data. Thus interactivity on single pages is assured. To mark potential threats on different sub-pages, the established icons and colors could be used.

6.4 Analysis Strategies and Usage Scenario

To evaluate results from the anomaly detection in Figure 6.5 on the following page, our visualization system was employed. The considered data set contains device readings from a modern six-stage process of water treatment. The monitored sub-

processes are:	P1	Raw water storage
	P2	Pre-treatment
	P3	Membrane Ultra Filtration (UF)
	P4	Dechlorination by Ultraviolet (UV) lamps
	P5	Reverse Osmosis (RO)
	P6	Disposal

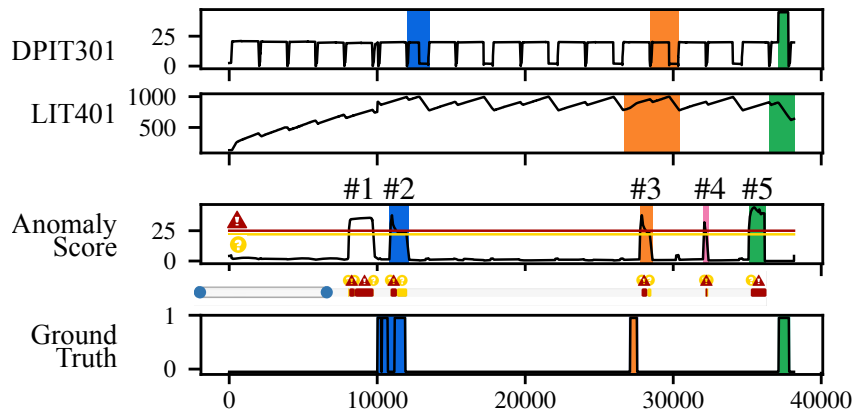


Fig. 6.5.: Anomaly detection results (extended Figure 6.2 on page 107): Ground truth is provided and detected anomalies are highlighted.

Connections between the sub-processes are shown in Figure 6.6.

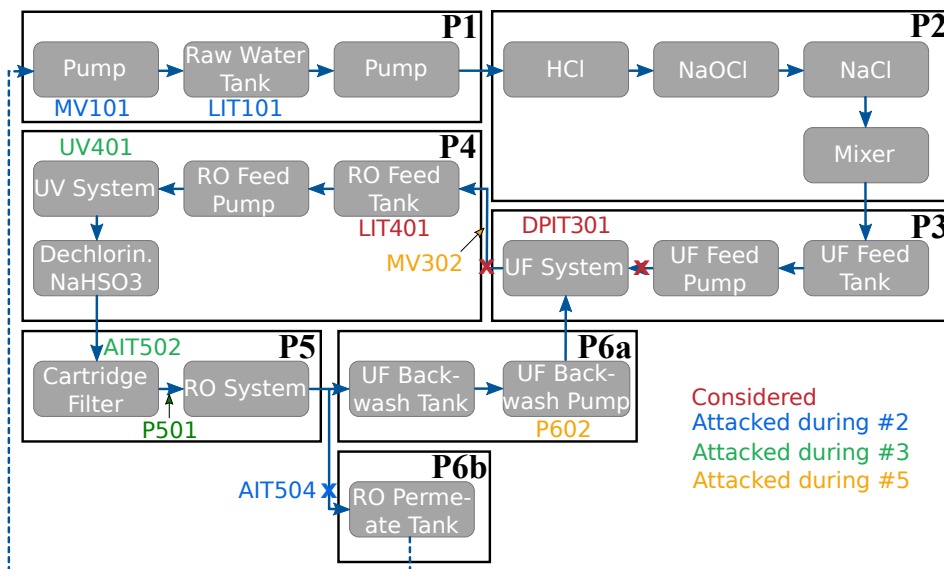


Fig. 6.6.: Sub-processes of the used data set and their interplay: Attacked components and the sensor and actuator considered in the usage scenario are highlighted.

The digital part of the system consists of a layered OT-network, PLCs, human machine interfaces and a *Supervisory Control And Data Acquisition* (SCADA) workstation. PLCs are industrial digital computers that are tailored to controlling manufacturing processes and can be connected to devices. In total, 51 devices are controlled by the PLCs. A detailed list including a description was given by Goh et al. [70]. Here, also the attacks contained in the data set and their occurrences are described.

The differential pressure indicating transmitter controlling the back-wash process in stage P3 (sensor DPIT301), and the actuator LIT401 reporting the water tank level

of the tank in stage P4 are considered in this usage scenario. See Figure 6.6 on the facing page for the location of these components in the overall process. The analyzed data consists of 10,000 samples of normal behavior as a “base line” to compare occurring patterns to, and 28,000 samples containing five attacks. Five anomalies have been detected in the readings of sensor DPIT301. For actuator LIT401, just one anomaly, coinciding with the first anomaly in DPIT301, was detected (not shown here). This first detection is an artifact at the edge between normal data and data containing attacks. Furthermore, the 4th detection in the readings of DPIT301 is a false positive. In Figure 6.5 on the preceding page, the detected anomalies and corresponding areas in the device readings are highlighted. In addition, the ground truth is given, where value 1 stands for an attack. For every anomaly, a different analysis strategy that is supported by our visualization tool is applied. The five exemplary situations are:

- **Period disruption:** a sudden change in the period of the measurements occurs.
- **Abnormal occurrence of values:** values that are in the normal range occur at an abnormal time point or for an abnormal length.
- **Phase shift:** the period of the measurement changes.
- **False positive**
- **Abnormal values:** values outside the normal range occur.

The illustrations in this section are reproduced from [*116].

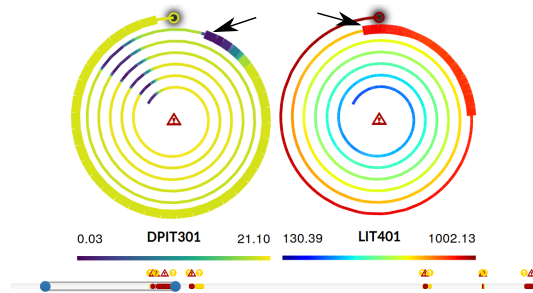
Detected Anomaly #1, Period Disruption. This detection is caused by the edge between normal data and data containing attacks. The combination of two data sets caused a period disruption and a sudden change in values.

In our visualization system both effects are visible: The period of DPIT301 in the normal data previous to any attacks was perfectly identified by the period estimation (Figure 6.7a on the following page). Widening the considered time interval to include the detected anomaly provides insight in the reasons for the detection (Figure 6.7b on the next page): Clearly, the period of DPIT301 is disrupted in the area that was detected as abnormal (left). In addition, the small, abrupt increase in the values of LIT401 is visible, especially if the jet colormap is chosen (right). Both abnormalities are easily **separated** from normal readings by **comparing** the readings.

Detected Anomaly #2, Abnormal Occurrence of Values. Three short attacks in rapid succession lead to this abnormal behavior. Two attacks were executed on permeate



(a) The period of DPIT301 before the first anomaly.



(b) The anomaly occurs: (left) Disrupted period, (right) abrupt increase.

Fig. 6.7.: Anomaly #1.

conductivity analyser AIT504 measuring the NaCl level in sub-process 5 at the reverse osmosis system. The expected outcome was missed and the attacks did not lead to much change in the data. The third attack was on the motorized valve MV101 that controls the water flow to the raw water tank at the beginning of the process and level transmitter LIT101 controlling the raw water tank level. The valve is kept on, while the transmitter is kept on a constant level, avoiding a shutdown of the valve. This leads to an overflow of the raw water tank and propagates further through the system, causing lagging.

This lagging is clearly visible using our system: After anomaly #1, the readings of DPIT301 re-start their periodical behavior with a phase shift. After one cycle, the next anomaly is detected, resulting in an extraordinarily long phase with low values that can be **separated** from the periodical behavior. Although the new period is not yet well established, this phase clearly stands out from the data and is obviously the reason for the detected anomaly (Figure 6.8). Hence the visualization leads to the correct classification of the detected anomaly as an attack.

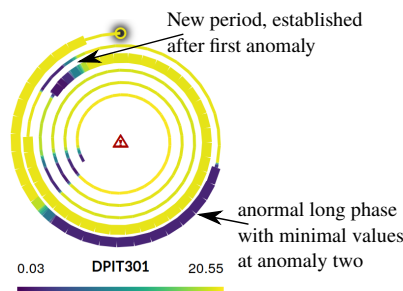


Fig. 6.8.: Anomaly #2: An extraordinarily long phase with minimal values for DPIT301.

Detected Anomaly #3, Phase Shift. Here, the abnormal behavior was triggered by an attack on the dechlorinator UV401 in sub-process 4, used to remove the chlorine

from the water. It was stopped and the value of oxidation-reduction potential analyser AIT502 monitoring the NaOCl level in the reverse osmosis feed was fixed to prevent an alert. In addition, the pump P501 pumping dechlorinated water to the reverse osmosis system was kept on. During the attack it was not possible to force P501 to stay on; so a possible damage was avoided and the chlorine loaded water was rejected at sub-process 6. This leads to higher input in the ultrafiltration process which is visible in the readings of DPIT301 and LIT401 using our system after an adaption of the period:

After anomaly #2, the period of DPIT301 from the second data set containing the attacks is fully established. The estimated cycle length is no longer valid and the periodicity is not easy to spot. The cycle length of the visualization can be easily adapted by **combining** similar readings, so that a further analysis of the period is possible (Figure 6.9).

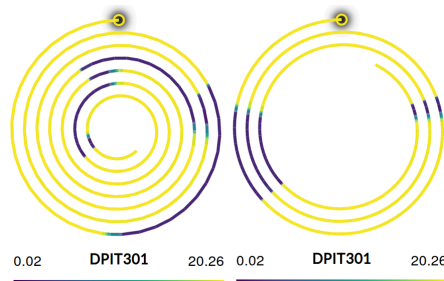


Fig. 6.9.: Cycle length adjustment: (left) Automatically detected period length, (right) after manual adjustment.

At detected anomaly #3, periods of DPIT301 and LIT401 change the phase shift. Although in LIT401 no anomaly was detected, this shift is clearly visible in our visualization system after adapting the cycle length (Figure 6.10). This identifies the detected anomaly as an attack (which is confirmed by the ground truth). The phase shift of LIT401 is also difficult to spot in Figure 6.5 on page 114, leading us to the impression that our visualization system is indeed superior to naïve time series visualizations, supporting a more detailed **comparison** of device readings.

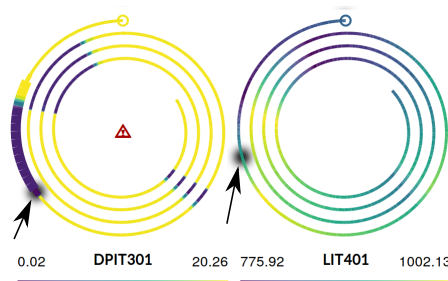


Fig. 6.10.: Anomaly #3: Period shift.

Detected Anomaly #4, False Positive. Around detected anomaly #4, no changes in values or periods are visible. Hence, an attack at this time point is doubtful. In fact, this detected anomaly is a false positive that can be identified fairly easy using our visualization system (Figure 6.11), **comparing** the readings in the interval to normal behavior.

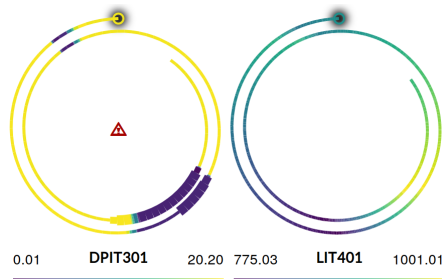


Fig. 6.11.: Anomaly #4 is a false positive. No abnormal behavior can be spotted visually.

Detected Anomaly #5, Abnormal Values. The triggering attack for this anomaly detection involved keeping the backwash pump P602 in sub-process 6 closed, setting the value of DPIT301 to a high value and keeping the motorized valve MV302 that controls the flow from the ultrafiltration process to the de-chlorination unit open. This leads to a system freeze since no water from the backwash pump is available while water transport from sub-process 3 to 4 is kept active. The reverse osmosis feed tank runs dry.

Again, this attack is clearly visible in our system. Even tracing the origin of the attack is possible: At the time frame of detected anomaly #5, the values of both devices change tremendously and fast, indicating an attack. Choosing the colormap relative to the current time frame, this is easy to see for both devices making it easy to **combine** both as affected by the attack, and **separate** the anomaly from normal behavior (Figure 6.12 on the next page).

The direct attack on DPIT301 and also the drop in values in LIT401 caused by the dry running tank are clearly visible. In addition, the succession of visible effects can be traced using the spotlights to **compare** and **combine** sensors. It indicates that the attack first affected DPIT301 and propagated to LIT401 afterwards. This provides a hint on the origin of the attack. At the beginning of the manipulation of values in DPIT301, the anomaly detection provides values from category II, as the value remains unusually high, category III is detected. This increase in values (and thus line thickness) is clearly shown and **separated** from normal behavior. Hovering

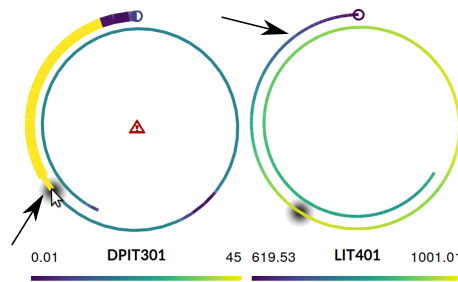


Fig. 6.12.: Anomaly #5: (left) sudden increase in DPIT301, (right) rapid decrease in LIT401.

the alert icon at the spiral center, the segment of category II will be highlighted in yellow, the segment of category III in red.

Overall, our visualization system proved useful in supporting cyber security experts in their triage analysis tasks. Feedback of the experts is given in Section 6.5. User studies for this system and a further developed version described in Chapter 8 are given and discussed in Chapter 9.

6.5 Expert Evaluation

To evaluate usability and usefulness of our visualization system for cyber security experts, an expert evaluation was performed. We interviewed an expert in the context of a presentation and application of our system. Considering usability and visual presentation of the application, the expert's opinion was very positive. In his judgement, identification of correctly and incorrectly detected anomalies was easy to accomplish using our system. Especially the different employed highlighting techniques for anomalies and potential anomalies were convincing. Also the overall data representation in his opinion provided a good overview and the maximal length of the visualized time frame is well chosen. The expert stated that our system could be beneficial not only in an industrial cyber security context, but also for process-level monitoring in industrial applications when used to augment existing monitoring systems.

As an enhancement of our system, the expert suggested the possibility to not only visualize continuous time frames, but to allow comparison of discrete time frames. An application example would be periodic behavior of a process that is not on an hourly scale but develops within several days. Furthermore, recurring attacks performed by staff or external employees are conceivable. These would occur only

during certain times of the day, or even at certain days of the week. Hence, providing the possibility to show spiral plots not only per device but also per selected time frame is an option for further development.

6.6 Discussion and Conclusion

Our Security in Process System supports triage analysis and monitoring of OT networks. It exploits typical patterns that are often inherent in device readings from industrial processes to enable cyber security experts and laymen to perform triage analysis and monitoring of the system simultaneously. The main characteristic of our visualization system are manipulable spiral graphs that combine the visualization of device readings in their coloring with the results from anomaly detection in their line thickness. Anomalies are highlighted using further pre-attentive properties like form, movement and dedicated colors.

In the example usage scenario, we found that triage analysis using our tool was effective, comfortable and superior to naïve time series visualizations. It also supports the previously defined basic visualization tasks. Each correct anomaly detection could be explained by different, clearly visualized features in the data. Missing these features for the false positive easily leads to rejecting the detected anomaly. Being able to switch the colormap boundaries from “relative to the whole data set” to “relative to the current time frame” turned out to be helpful when considering either long-term development in the data or local behavior, respectively. Clear accentuation of anomalies using icons, line thickness and highlighting with determined colors leads to an easy to understand overview of the data and potential anomalies.

Options for further development of the system are the the automated proposition of meaningful time frames based on the results from anomaly detection. Also, a referenced visualization of physical device locations in the machines to support orientation and consequently the meaningful division of the devices in groups could increase the overview, especially when many devices are considered at once. Moreover, embedding operational constraints in the visualization could be possible, e.g. by putting device visualizations into context of an annotated layout plan containing these constraints. Also, the enhanced analysis of inter-device relations is currently only supported by the spotlight feature and could be more in focus of the system.

Based on user and expert feedback, we furthermore identified collaboration via the system as an important addition to user support. Consequential, we developed the *enhanced Security in Process System* that incorporates knowledge from a common knowledge base in the visualization to allow collaboration and support between users. The enhancement of the spiral plots also provides the opportunity to compare multiple separate anomalies from separate time windows, as encouraged in the expert interview. To enhance our system, we developed and used the *Knowledge Rocks Framework* described in the following Chapter 7. Its application to the Security in Process System is described in Chapter 8. Both, the Security in Process System and its enhanced version are evaluated in user studies in Chapter 9.

Knowledge Assistance for Decision-Making

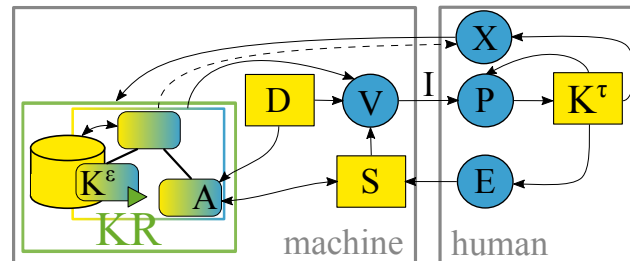


Fig. 7.1.: The **Knowledge Rocks Framework** provides all components that are required to make a visualization system knowledge-assisted according to the KAVA model.

Knowledge-assisted visualization systems use stored knowledge to support analysis tasks. The stored knowledge can range from different visualization strategies over best practices in a single visualization system to specific application knowledge. The provided support can be implemented as guidance regarding visualization settings or choices, automated analysis and pre-processing of the input, generation of examples and much more. For example **comparing** data not only to other readings in the same data set, but to stored readings from the knowledge base, **combining** previously analyzed results with current ones and **separating** behavior that is not part of the knowledge base allows a deeper understanding of the data.

The utility of knowledge-assisted visualization systems is well understood: already in the late 90's, Fujishiro et al. developed the GADGET system that supports users in choosing suitable visualization systems for their goals under given constraints (e.g. data properties) [63]. Since then, the idea of knowledge assistance was further augmented and formalized. Pike et al. declare knowledge-based interfaces as an important research challenge [154], and knowledge-assisted visualization was discussed in an issue of the *IEEE Computer Graphics and Applications* journal [42].

Also in the collaborative setting, knowledge-assisted visualization systems exhibit their strength: users are able to share their knowledge between each other, even if they are not using the system concurrently. Collaborative and cloud based visualization systems both have access to the knowledge of multiple users and are able to spread it among their users with a single, central knowledge base.

Despite the usefulness and increasing necessity of knowledge assistance in visualization systems, to our knowledge there are no general frameworks for its realization. Specific implementations pursue different approaches and are not generally applicable. Theoretical approaches, on the other hand, provide impulses concerning the development of new knowledge-assisted systems, but significant transfer must be performed on an ad hoc basis to translate theory into concrete implementations. Finally, fully implemented frameworks that add knowledge-assistance to visualization systems limit the possibilities of knowledge integration and interaction with the knowledge base.

Hence, while incorporating knowledge assistance is a helpful enhancement for many visualization systems, there is only few support for the enhancement of an arbitrary system. To not only enhance the Security in Process System to be knowledge assisted, but also facilitate the enhancement of other visualization systems, I developed the Knowledge Rocks Framework together with Dr. Simon D. Duque Anton, Prof.Dr. Christoph Garth and Prof.Dr. Heike Leitte. Christoph Garth and Heike Leitte supported me in developing the framework, Simon D. Duque Anton provided useful information on machine learning approaches for the application of the framework to the Security in Process System (Chapter 8). With the Knowledge Rocks Framework, we aim to bridge theory and practice of knowledge-assisted visualization by simplifying the extension of existing visualization systems to become knowledge-assisted, allowing an effective reactivation of software resources in the visualization community. We published the framework at IEEE Vis 2021 [*115].

I present our Knowledge Rocks Framework in Section 7.2, with the derivation of its requirements (7.2.1) and definition (7.2.2). Validation of its generality and applicability is provided in Section 7.3, and its application to several examples is outlined in Section 7.3.1. Chapter 8 then describes in detail its implementation in the enhanced Security in Process System.

7.1 Review of Knowledge Assistance in Visualization

Knowledge-assisted visualization aims to incorporate knowledge into the visualization process to support users [132]. Chen et al. provided a solid theoretical basis [40, 41, 132] and Federico et al. developed a conceptual model of knowledge-assisted visual analytics – the KAVA model [57].

Around this basis, a particular taxonomy evolved. We give a short overview of important terms.

Tacit knowledge contains a user’s personal knowledge about data and insights gained during the perception of the visualization [57]. Knowledge that is available to assist the visualization forms the *knowledge base*. To incorporate tacit knowledge into the knowledge base, it needs to be “written down” and become *explicit knowledge*. This process is referred to as *externalization*[144, 205]. Besides *direct externalization* where a user is explicitly writing down their knowledge, *automated externalization* methods can be used that continuously extract knowledge in the background, for example from user interactions (*interaction mining*). Different types of knowledge provide different support when they are leveraged in knowledge-assisted visualization: *Operational knowledge* is about handling the visualization system and supports users in interacting with the visualization. *Domain knowledge* contains knowledge about the analyzed data and helps users to interpret the content of the visualization. Domain knowledge splits further in *declarative knowledge* explaining *what* can be seen in the visualization and *procedural knowledge* about *how* to react to the data and how to make decisions and take action in the application domain.

7.1.1 The KAVA Model

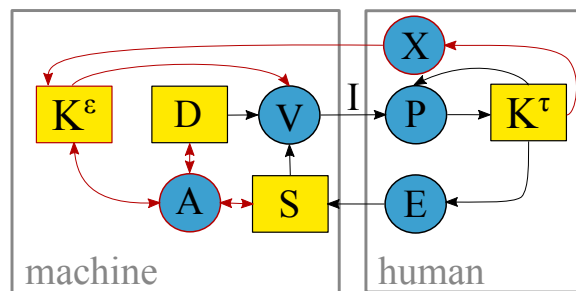


Fig. 7.2.: The KAVA model of knowledge-assisted visual analytics by Federico et al. [57]. The processes: Analysis (A), visualization (V), externalization (X), perception/cognition (P) and exploration (E); the containers: Explicit knowledge (K^e), data (D), specification (S), tacit knowledge (K^t), and a non-persistent artifact: image I. Edges describe potential influences of data or knowledge stores and processes. Components and influences that distinguish a visualization system from a knowledge-assisted one are highlighted in red.

The *KAVA model* by Federico et al. (Figure 7.2) incorporates an explicit knowledge store and several knowledge-related processes in van Wijk’s operational model of visualization [212]. The basic knowledge assistance-related processes that are represented by their model are:

Knowledge Visualization $\boxed{K^e} \rightarrow \boxed{V}$. Visualization \boxed{V} of explicit knowledge $\boxed{K^e}$ to present e.g. automatically extracted knowledge.

Simulation $\boxed{K^e} \rightarrow \boxed{A} \rightarrow \boxed{D}$. External knowledge $\boxed{K^e}$ is analyzed \boxed{A} and applied to generate new data sets \boxed{D} that provide users with supporting scenarios.

Automated/Intelligent Data Analysis $(\boxed{D}, \boxed{S} / \boxed{K^e}) \rightarrow \boxed{A} \rightarrow \boxed{K^e}$. The application of automated analysis \boxed{A} to data \boxed{D} to generate explicit knowledge $\boxed{K^e}$, given a certain specification \boxed{S} or using explicit knowledge $\boxed{K^e}$ respectively.

Direct Externalization $\boxed{K^t} \rightarrow \boxed{X} \rightarrow \boxed{K^e}$. The system supports users in actively formulating tacit knowledge $\boxed{K^t}$ through an appropriate direct externalization interface \boxed{X} to obtain explicit knowledge $\boxed{K^e}$.

Interaction Mining $\boxed{K^t} \rightarrow \boxed{E} \rightarrow \boxed{S} \rightarrow \boxed{A} \rightarrow \boxed{K^e}$. Using their tacit knowledge $\boxed{K^t}$, a user explores \boxed{E} the data. The interaction with the system results in different specifications \boxed{S} over time. These specifications are “mined”, that is they are automatically analyzed \boxed{A} and contribute to the knowledge base $\boxed{K^e}$.

Guidance $\boxed{K^e} \rightarrow \boxed{A} \rightarrow \boxed{S}$. Explicit knowledge $\boxed{K^e}$ is analysed \boxed{A} and used to guide the user’s choice of settings \boxed{S} . As described by Ceneda et al. [36], there are different degrees of guidance: visual cues, providing alternative options, and prescribing specifications. An example for visual cues is the marking of abnormal high values in our exemplary implementation in Chapter 8.

The KAVA model aims to inspire innovative design approaches for knowledge-assisted visualization systems. Yet, creating such a system based on the model requires substantial transfer from the theoretical model to a practical implementation. With the Knowledge Rocks Framework, we lower this hurdle. To do so, we propose an application-agnostic architecture built around an ontology definition that, integrated in a visualization system, adds all functionality to the system to become knowledge-assisted in terms of the KAVA model.

7.1.2 Knowledge-Assisted Visualization Systems

There is a large variety in knowledge-assisted visualization systems that have been developed so far and are described by the KAVA model. In the following, we present a representative selection of recent systems to illustrate the wide range of applications. We furthermore use the presented systems as running examples to illustrate requirement analysis and application of the Knowledge Rocks Framework.

◆ *VUMO: Towards an Ontology of Urban Mobility Events for Supporting Semi-Automatic Visualization Tools* [181] – Sobral et al. describe two ontologies that formalize the knowledge related to urban mobility events and visualizations respectively. They use this knowledge base to propose appropriate visualization techniques. Furthermore, they provide a pipeline for a user-centered design process of visualization tools.

◆ *Knowledge-Assisted Comparative Assessment of Breast Cancer using Dynamic Contrast-Enhanced Magnetic Resonance Imaging* [143] – Nie et al. support physicians in exploration and classification of breast lesions. Lesions get scored based on their cluster structure and a knowledge base containing previously classified lesions using a fuzzy inference system.

◆ *Formalizing Visualization Design Knowledge as Constraints: Actionable and Extensible Models in Draco* [137] – Moritz et al. implement a knowledge base containing visualization systems as well as hard constraints and weighted soft constraints concerning their application. Aiming on the acceleration of the transfer of research knowledge into practical tools, Draco guides users concerning visualization settings and choices.

◆ *KnowledgePearls: Provenance-Based Visualization Retrieval* [184] – Stitz et al. propose a system to build and visually access a provenance graph that stores visualization states and actions that occur during data analysis. They support direct access to the provenance graph based on queries (by selecting properties, and formulating statements in system or natural language) or based on created examples.

All of these systems support data analysis: By proposing an adequate visualization (◆,◆), by proposing visualization parameters (◆) and by intelligent data analysis (◆). A survey of 32 additional systems was given by Federico et al. [57].

Ontologies. In computer science, *ontologies* are a form of knowledge representation. Drawn as graphs, nodes of ontologies with degree one represent instances and nodes with higher degree represent classes or concepts. Edges represent the relationships among the classes with the most common relationship being “is a”. Ontologies are

ideal to store knowledge about whole classes (*class-based* knowledge) as opposed to *case-based* knowledge about specific instances. They make stored knowledge accessible for both –humans and computers– because of their hierarchical structure. In addition, they have the ability to mutate as additional knowledge gets available.

According to Carpendale et al. ontologies will be indispensable in developing infrastructures for knowledge-assisted visualization [30]. This application is discussed for example by Miksch et al. [132]; examples are the ontologies defined by Sobra et al. to support integration and visualization of data from intelligent transportation systems [181] and the three ontologies used by Gilson et al. to determine an appropriate visualization for web data (domain, visual representation and semantic bridging ontology) [67].

In our Knowledge Rocks Framework, the ontology is the starting point for an existing visualization system to become knowledge-assisted. It is used for structured storing, processing and retrieving of knowledge.

We anticipate that knowledge-supported visual analysis will play a seminal role in future visualization systems and provide with the Knowledge Rocks Framework support for the effective reactivation of existing visualization systems. To demonstrate the high flexibility and generality of the proposed ontology-centered architecture, we motivate our design choices based on the running examples in Section 7.2.1 on the following page, and sketch their possible implementation based on the Knowledge Rocks Framework in Section 7.3.1 on page 136.

7.2 The Knowledge Rocks Framework

The Knowledge Rocks Framework provides a process to incorporate knowledge-assistance in a visualization system. It centers around an application-agnostic architecture that includes the existing visualization system as one of three components. Thus, the abstract process of knowledge incorporation boils down to the concrete implementation of components and establishing their interaction. Within this implementation, all processes of a knowledge-assisted visualization system –as described by the KAVA model– can be reproduced. To become a component in an implementation that bases on the Knowledge Rocks Framework, an existing visualization system is only required to provide extensibility for the integration of the knowledge-assistance and interfaces for component interaction.

7.2.1 Requirements

The KAVA model is a general model for knowledge-assisted visual analytics. Hence, it determines components that need to be added to a visualization system to make it knowledge-assisted. These components are: explicit knowledge stored in a *knowledge base* K^c and *automated analysis* \textcircled{A} using this knowledge base. An optional addition is support for *direct externalization* \textcircled{X} . Required and optional components are colored in red in Figure 7.2 on page 124.

We examined several knowledge-assisted visualization systems with respect to their specific implementation of these components to obtain an application-agnostic architecture with a wide application range. In the following, we exemplarily discuss the running examples given in Section 7.1.2 on page 126:

◆ VUMO's knowledge base consists of two ontologies –a characterization of data and a characterization of visualizations– and all integrated visualization systems and data sets. New data sets or queries are automatically analyzed by classifying them using the data ontology. The assigned perception factors are then used to link and suggest visualizations from the knowledge base. VUMO supports direct externalization by domain experts via the data ontology, and by visualization experts via the visualization ontology and the proposed visualization development pipeline.

◆ The knowledge base proposed by Nie et al. contains scored lesions with their linked clusters of time intensity curves, and linguistic rules that are created for each stored lesion. The main focus of this system lies on the automated analysis of dynamic contrast-enhanced magnetic resonance imaging data sets consisting of time intensity curve extraction, clustering, and scoring of lesions. Externalization of expert knowledge is supported by direct access to the knowledge base: scored data is presented together with stored results to support a possible correction of the scoring. Also, the human-readable linguistic rules for the scoring are available to users.

◆ Draco's knowledge base consists of specifications of visualization systems and constraints. Its scope can be adapted depending on the application: Different visualization types or different specifications for a single visualization type can be stored. The input data –data properties and an incomplete specification of the visualization– is used to automatically suggest an optimal visualization from the knowledge base using cost functions with (optionally learned) weights. Hence, the automated analysis of the data is the optimal completion of the specification. Direct externalization by visualization experts is supported by providing a syntax to describe visualizations and constraints.

◆ The knowledge base of KnowledgePearls consists of a provenance graph together with stored properties. These properties are currently considered as independent; however, the authors state that additional knowledge concerning their dependencies could be integrated, which is part of their future work. The focus of KnowledgePearls lies on the direct access to the knowledge base, which is implemented with different query options. Thus, automated analysis of user interactions is kept simple: they are added to the provenance graph. The authors state that the size of the provenance graph is a limiting factor for externalization support.

We extracted the following common patterns with respect to the required KAVA processes:

In all systems, the *knowledge base* consists of two parts: A “passive” part with concrete instances (visualization systems in ◆ and ◆, lesions in ◆, provenance in ◆) and an “active” part consisting of rules or constraints acting on these instances (ontologies in ◆, linguistic rules in ◆, constraints and cost functions in ◆, stored properties and their dependencies in ◆). This active part contains general relationships and reasoning used to classify input data. It corresponds to the “concepts” component in the structure defined by Rind et al. [162].

In the first three examples, *automated analysis* is implemented as the action of rules or constraints on the analyzed data and the knowledge base: ◆ – linking to stored visualization systems using the ontology, ◆ – automated lesion scoring based on stored lesions and linguistic rules, and ◆ – completion of the incomplete specification using constraints, cost functions and stored specifications.

In ◆ KnowledgePearls, the stored properties are currently not subject to constraints. Thus, the automated analysis process does not rely on any rules or constraints. On the other hand, KnowledgePearls is the only system among the considered examples that focuses on direct access to the database. This access –via query, by definition or example– follows the same pattern as the automated analysis in the other systems: given (and potentially stored) rules and constraints are applied to the knowledge base.

These observations –the knowledge base consisting of a passive and an active part where the active part acts on the passive instances to provide automated analysis or direct access– together with the structural definition of required and optional components in the KAVA model lead us to the following architecture definition.

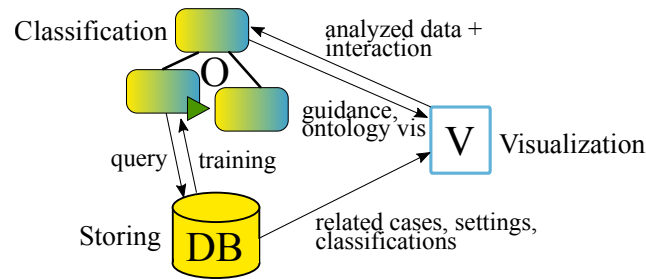


Fig. 7.3.: Data flow between the three parts of the Knowledge Rocks Framework. Two components are integrated in the visualization system (V): the acting ontology (O) that automatically classifies the analyzed data and the database (DB) that stores ontology instances and related structures.

7.2.2 Architecture Definition

The proposed architecture consists of two parts that are integrated into the existing *visualization system*: an *acting ontology* and a supporting *database* to store the ontology's instances and related data structures. See Figure 7.3 for their interaction.

This structure implements the proposed structure by Rind et al. for domain knowledge in visual analytics [162]. Concepts are stored in the acting ontology and data sets in the database. The fulfillment of the desiderata they identified for structural models of domain knowledge are either inherent in our system (knowledge should be machine interpretable, pre-existing taxonomies can be used, implementation of the knowledge sources proposed in [57], human readable knowledge, facilitating the exchange between different systems) or depend on the specific implementation (focussing on domain knowledge, compatibility with heterogeneous data, standardized form to include provenance information e.g. using triggers in the database) and the chosen language or tool for this implementation (software library support).

In the following we present the different parts of our framework, give implementation and integration details, and discuss advantages and design choices.

Acting Ontology. As the centerpiece of the resulting Knowledge Rocks Framework, we define an acting ontology (O in Figure 7.3) as an ontology with callback functions. These callback functions are defined for every class of the ontology and allow automatic traversal by deciding based on input data to which neighbor of the corresponding class the traversal proceeds. Hence, an acting ontology allows automatic analysis and classification of input data.

Callback functions are formulated by domain experts and can include procedural reasoning, machine learning and other structures that decide which path of an

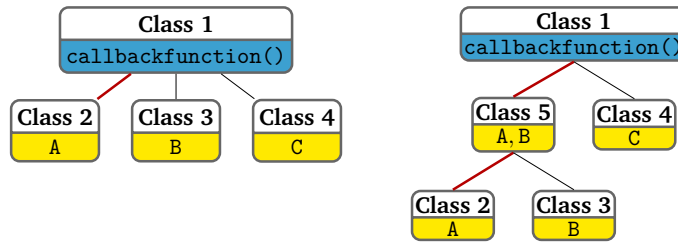


Fig. 7.4.: The **acting ontology**: The input data is classified as Class 2 if the callback function returns A. right: In case of a callback function that classifies more in-depth than the direct children, a list of possible results that is stored in a child’s properties determines further traversal.

ontology to follow, and which thereby establish parent-child relations between the nodes. By traversing the acting ontology to an instance, the input data is classified as belonging to this instance.

Concerning the concrete implementation of the acting ontology, we suggest the following: the ontology must to be interpretable by a computer; hence we implemented it in the web ontology language OWL [15]. Paths in the ontology are coupled to results of the callback functions using ontology properties (cf. Figure 7.4): a class of the acting ontology has a callback function as property. Every child of this class has a property listing one possible output of its parent’s callback function, determining which child is the next class on the path. In some cases, a callback function is able to classify more in-depth than its direct children. In this case, a child class might not contain a callback function but a list of its children’s properties. A concrete implementation is discussed in Chapter 8.

We chose the ontology based approach for the following reasons: ontologies are a useful tool to capture and externalize knowledge. Furthermore, their structure allows them to capture the active part of the stored knowledge. They are able to capture both, quantitative and nonquantitative knowledge, which is for example useful in Biology which is “notoriously nonquantitative” as pointed out by Carpendale et al. [30]. Also, they impose minimal boundaries to the designer who is able to choose classes, instances and their relations freely and still make the stored knowledge machine readable. The ability of ontologies to support knowledge externalization can be derived from their proximity to learning and brain storming techniques like mind maps [29] and –even closer– concept maps [146]. There is an active research field treating knowledge externalization techniques including ontologies: for example Ishikawa et al. propose an ontology based approach for knowledge externalization in companies [90], and Aranda-Corral et al. developed a tool that allows the collaborative development and improvement of an ontology [8]. All

results from this area can be applied to design the ontology of the Knowledge Rocks Framework and develop it further.

Our choice to couple the ontology with callback functions to make it an acting ontology bases on several aspects: to allow automated analysis and direct access to the knowledge base by applying the “active” part of the knowledge (stored in the ontology) to the stored instances, an acting part with executable rules and constraints is required. By incorporating this acting part directly in the ontology, we avoid a gap between concept and implementation. Knowledge by domain experts that is captured in the ontology is directly part of the knowledge-assisted visualization system. The most common relation between two classes in ontologies –“is a”– is in most cases easily translated in a callback function that tests if the given input “is an” instance of the different descendant classes. Designing more elaborate callback functions on the other hand requires the identification of logical flow between the defined classes and thus provides additional support for knowledge externalization. Finally, the acting ontology is machine *and* human readable and thus presents a piece of self-documenting code that can further support knowledge externalization (for example the user classification of instances that fail traversing the ontology) and the understanding of the visualization system.

Database. The database (DB in Figure 7.3 on page 130) stores the “passive” instances and is tightly coupled to the acting ontology by the classification of instances as classes that are contained in the ontology. The nature of these instances is defined in the specific implementation, but we suggest to focus on domain knowledge as proposed by Rind et al. [162]. Whether the visualized data and the knowledge base reside in the same database is a design choice – depending on the stored knowledge, having both in the same database can avoid data amplification. Instances are stored automatically with all required additional information, after their classification by traversing the acting ontology. Hence, analyzed data that has been classified can be linked to data examples of the same class from the database. Stored data examples can contain operational data to support the visualization e.g. “All instances of the class ‘Contains Vortex’ are visualized using streamlines”. They also can represent domain knowledge, e.g. typical patterns in the analyzed data. See our enhancement of the Security in Process System in Chapter 8 for examples.

The database is required since our system is intended to potentially store numerous instances. Storing them (and related data) in the acting ontology would make them difficult to access. In addition, a lot of background information and data points might be stored for an individual instance, blowing up the acting ontology. Databases are

the intended tool for such applications, providing easy access to customizable data structures. The link to the acting ontology is given by the stored classification, that is the parent class in the ontology structure.

Visualization System. Finally, both entities –acting ontology and database– need to be coupled to the given visualization system (V in Figure 7.3 on page 130). Besides automatic analysis that takes place in the background, there are two main types of interaction between users and the knowledge base in a knowledge-assisted visualization system: receiving guidance from the system and direct access to the knowledge base via knowledge visualization (as described in the KAVA model, Section 7.1.1 on page 124). In our framework, both types rely on the acting ontology.

Guidance can help to narrow a user’s knowledge gap by building upon various inputs like data, interaction history, stored domain knowledge and more [36]. The different guidance approaches –visual cues, providing alternative options, and prescribing of specifications– can either depend on a whole class of instances (for example a visual cue: “All instances of the class ‘Abnormal high values’ are visualized with a cue in the respective area”), or depend on individual instances (for example prescribing of specifications: “This specific example is visualized with a period of 3.5”). In the first case, they are triggered directly by the classification via the acting ontology: as soon as the provided data is classified as a class with a property that triggers guidance, this trigger is passed to the visualization system and processed. In the latter case, guidance is offered if the instance that triggers the guidance is selected. An example for this is the optional prescription of stored specifications in our implementation in Chapter 8.

Visual cues and alternative options require enhancements of the existing visualization system. While visual cues are incorporated directly in the visualization, alternative options are visualized stand alone, in a similar fashion as the analyzed data to allow a comparison. A straightforward possibility is to use the existing visualization not only for the analyzed data but also for related instances. Then, both visualizations can be combined using for example juxtaposition or superposition. Of course, more elaborate or application specific techniques can be applied and are in many cases preferable. A survey on comparative techniques in information visualization was given by Gleicher et al. [69]. They also provide a design strategy for comparative visualization [68]. In Scientific Visualization, comparative visualization of three-dimensional ensembles was addressed by Demir et al. [47], Zhang et al. address tensor fields [220] and Verma et al. present comparative flow visualization [198].

Direct access to the knowledge base requires structures that allow filtering and querying of the stored data. Allowing to choose classes and paths in the acting ontology provides these structures. Using an interactive ontology visualization, users are able to browse different instances in the database by choosing classes until only currently helpful instances remain. The user's decision which path to take in the acting ontology can be supported by comments in the callback functions describing the reasoning behind the functions. These comments can be made accessible using a documentation generator. Furthermore, browsing the acting ontology can support users in manually classifying instances if the automated classification fails for some reason. There are multiple ontology browsers with different visualizations available: for example jambalaya [186] and OntoViz [180]. A recent survey on this topic was given by Dudas et al. [50].

An example for an acting ontology with callback functions, a database structure and the integration in a visualization system is given in Chapter 8.

7.3 Implementation Steps for KAVA Processes with the Knowledge Rocks Framework

Via the step-by-step implementation of the Knowledge Rocks Framework in a concrete application, knowledge-assistance is added to the embedded visualization system.

1. *Specify the knowledge* that builds the knowledge base, identify classes and instances and build the ontology and the database structure.
2. *Add the active part* of the knowledge base: implement callback functions that allow a traversal of the ontology and hence automated classification and analysis.
3. *Integrate* the knowledge base into the visualization system.

There are many possible interactions and processes that are typical for knowledge-assisted visualization systems and that can be implemented with the Knowledge Rocks Framework. In the following, we sketch the implementation of knowledge-assistance-related processes as described in the KAVA model (Section 7.1.1 on page 124):

Knowledge visualization requires structures to browse the knowledge base. In the Knowledge Rocks Framework, the database can be queried by an interactive visualization of the acting ontology. Selecting specific classes, a user is able to restrict

the obtained results and search for useful instances that are classified as one of the selected classes. Besides automatically offered guidance by the system, this visualized acting ontology allows users to get an overview of the available data in the knowledge base. The visualization of the result is provided by the visualization system, either stand alone or illustrative, in addition to the visualization of currently analyzed data.

Automated and intelligent data analysis is performed by the acting ontology in form of automated classification. Processing and analysis steps are incorporated in the callback functions. To implement intelligent data analysis, the analysis and classification can be automatically improved by either changing the ontology or the callback functions, for example by training contained machine learning systems.

Direct externalization of tacit knowledge is supported during the implementation of the system by requiring the definition of the ontology and its callback functions. During operation of the knowledge-assisted visualization system, the ontology's structure, the documentation of its callback functions and the provided classifications further support knowledge externalization.

Interaction mining and other automated knowledge generation processes are possible by automating the collection of specification data. After the collection was triggered, the collected data is automatically classified by the acting ontology and stored in the knowledge base.

Guidance can be implemented as described in Section 7.2.2 on page 130, based on the classification of the currently analyzed data by the acting ontology. It can either be triggered by the class in the acting ontology directly, or by instances of the same class that are stored in the database.

Simulation is a priori not a purpose of the Knowledge Rocks Framework. We focus on the analysis of existing data, not on creating new data. Nevertheless, an extension of the acting ontology-idea could be able to provide this, for example by linking functions that are able to generate data.

Thus, embedded in the KAVA model (Figure 7.1 on page 122), the Knowledge Rocks Framework keeps nearly the same connections as the knowledge-assisted specific components $\boxed{K^\varepsilon}$ and \textcircled{A} in the original model (Figure 7.2 on page 124). The edge $\textcircled{A} \rightarrow \textcircled{D}$ is missing since the Knowledge Rocks Framework does a priori not support simulation. A symbolic edge from the acting ontology to the externalization process \textcircled{X} is added, emphasizing support for users in externalizing their tacit knowledge. This support is provided via visualization $\boxed{KR} \rightarrow \textcircled{V}$, but also by providing a common taxonomy for all users. The proposed architecture provides the structure to build

the knowledge-assisted visualization on. Possible interactions with the knowledge base in a concrete implementation strongly depend on the integration of the added components in the system and may differ substantially. Rind et al. give some typical approaches for the integration of knowledge in visualization systems [162] and discuss an example for knowledge assistance with direct knowledge access [200]. Ceneda et al. developed a framework to guide the development of knowledge-assisted visualization systems [37].

7.3.1 Application of the Framework

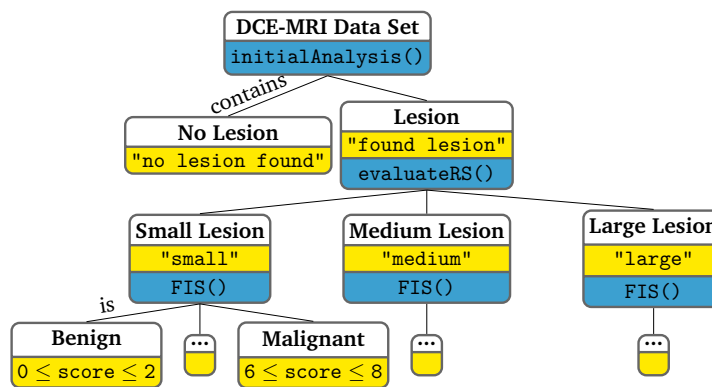


Fig. 7.5.: Sketch of a possible **acting ontology for** \blacklozenge : The classes correspond to the linguistic variables. The analysis of DCE-MRI data sets is represented in this acting ontology: after initial analysis –determining if there is a lesion detected or not– the lesion size is determined based on the relative size (RS). The lesion score is then based on the Fuzzy Inference System FIS. The visualization of this ontology has no correspondence in the original paper. Nonetheless, this would provide users a better understanding of classification options and boundaries in the system and a common terminology.

We briefly discuss possible implementations of the running examples using the Knowledge Rocks Framework, reproducing the functionality of the examples based on an acting ontology with attached database. The various natures of these examples demonstrate the wide range of results that can be achieved with the framework. A detailed case study of our application to the Security in Process System is given in Chapter 8.

\blacklozenge In VUMO, Sobral et al. define ontologies and discuss functions for automated classification, providing the callback functions in the acting ontology. The input data (a data set or the result of an analyzer) is classified by the data ontology resulting in a mobility event class, and spatial and temporal domains. With this classification set, the knowledge base can be queried to find an appropriate visualization based on the

defined human perception factors. Integration of visualization systems is done using the visualization ontology.

◆ The linguistic variables defined by Nie et al. provide the classes for an ontology definition (Figure 7.5 on the facing page). After the initial analysis of the data using clusters from the database, the classification of a lesion's size is done via evaluation of the relative size (RS). To further classify the result in terms of lesion scores, the linguistic rules are evaluated. The storage structure of lesions and clusters is easily implemented in a database. Using the classification by the acting ontology, the database can be queried for similar lesions with the same clusters and classification. The visual integration of the system is as described in the paper.

◆ With the knowledge base consisting of visualization specifications in Vega Lite syntax, a possible ontology for Draco is similar to a semantic tree of this syntax. An example for this and for possible hard and soft constraints is given in Figure 7.6. The callback functions deciding which soft constraints to follow or violate evaluate the cost functions. The ranked SVM discussed in [137] can be incorporated as such a callback function. Thus, the search implemented in Draco can be interpreted as depth first search in the tree structure of the ontology.

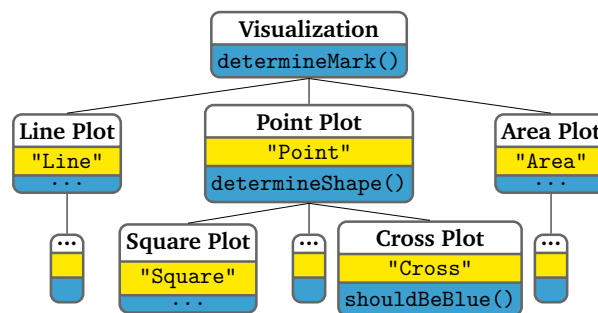


Fig. 7.6.: Extract of a possible **acting ontology** for ◆, following the semantic tree of the Vega Lite syntax. A hard constraint is that only point plots can have a shape. `shouldBeBlue()` is an example for a soft constraints for point plots with cross shape. This callback allows different paths, potentially generating costs when the constraint is violated. A depth first search in the complete ontology corresponds to Draco's search. Although it is not implemented in Draco, showing this ontology to the user would provide an overview of possible options.

◆ For KnowledgePearls, the interaction provenance graph represents the knowledge base and is already stored in a database. While structuring the properties in an ontology is still under research, a simple tree structure representing one property per level can be used instead of an ontology. With this tree structure, the classification and querying of mined interactions is possible. Querying using natural language or SQL syntax can be implemented on top of the ontology. Fuzzy search is a priori not implementable using our system since searching by classification gives absolute

```

1 def callbackfunction_left(data):
2     #based on data it is decided which neighbor of class A is the next class in the traversal
3     if data > 0.5:
4         return "A" #traversal proceeds with class 2
5     elif data > 0.8:
6         return "B" #traversal proceeds with class 3
7     else:
8         return "C" #traversal proceeds with class 4
9
10
11 def callbackfunction_right(data):
12     #based on data it is decided which neighbor of class A is the next class in the traversal
13     if data > 0.5:
14         return "A" #traversal proceeds with class 5, then further to class 2
15     elif data > 0.8:
16         return "B" #traversal proceeds with class 5, then further to class 3
17     else:
18         return "C" #traversal proceeds with class 4
19
20 #defining this dict, calling the defined functions is easily possible based on their name
21 callbacks = {}
22 callbacks["callbackfunction_left"] = callbackfunction_left
23 callbacks["callbackfunction_right"] = callbackfunction_right

```

Fig. 7.7.: Exemplary **callback function implementations** for the acting ontologies in Figure 7.4 on page 131.

results. The process can be mimiced by selecting multiple classes and rank the resulting instances based on the number of their ancestors that are selected. More details on this are given in the discussion of limitations (Section 7.5 on page 140). Query by example on the other hand is built in our framework: the user generated example is classified by the acting ontology and appropriate results from the database are presented.

The following tools can be used to implement the Knowledge Rocks Framework: The acting ontology can be created using Protégé [147] in Web Ontology Language (OWL). Software packages for loading OWL representations are readily available for a variety of environments. For example it can be loaded into Python using Owlready2 [109] and into JavaScript using owlreasoner [94]. To keep the acting ontology easy to read and to support the implementation and debugging of the functions, only the references to callback functions are given in the ontology properties. Hence, an additional file with callback implementations needs to be provided. An exemplary structure of such a file for the acting ontologies in Figure 7.4 on page 131 is given in the code in Figure 7.7.

7.4 Discussion and Limitations.

While the Knowledge Rocks Framework provides an uncomplicated starting point for the extension of visualization systems to be knowledge-assisted, there are still some hurdles that need to be cleared for such an extension and some limits in the framework itself.

The Ontology Structure. As the centerpiece of the Knowledge Rocks Framework, the ontology needs to be designed by experts and possibly with some effort. In some applications the structure of the knowledge to be stored is easy to find, but in other applications it might be a challenge to create an ontology structure. As a fallback, one can always use the tree structure representing one property per level instead of an ontology. Many callback functions need to be implemented in this case and browsing the tree to access the knowledge base is not as efficient as it could be with a carefully designed ontology. In addition, the design and implementation of callback functions potentially requires further expertise; however, we did not encounter substantially higher implementation complexity when using our framework compared to implementing the knowledge assistance from scratch. On the contrary, by directly incorporating results by domain experts, the implementation effort for developers can decrease.

Query Limitations. Since the Knowledge Rocks Framework bases on the classification of analyzed data and instances, fuzzy search with different keywords is not possible. Nonetheless, searching for multiple properties at once is possible by allowing the selection of multiple classes in the ontology. Ranking of the results based on the number of selected parent classes can provide similar results to a fuzzy search.

Since the classification is performed based on the ontology structure, it might be necessary to split especially real valued properties based on different value ranges. This prohibits searching for an exact value in the knowledge base. A solution for this is a second search in the search results of the ontology-query, searching for specific property values in the returned instances.

Required Extension of the Visualization. To be as general as possible, the Knowledge Rocks Framework gives a basic structure and leaves the integration in the visualization to the users. The visualization needs to be extended for example to

incorporate selected instances and to show instances from the knowledge base. Even the most simple approach using juxtaposition requires some implementation effort, especially if linking of the different visualizations is implemented. However, this effort is required also without using the Knowledge Rocks Framework.

Misleading Knowledge. Misleading knowledge is one of the biggest issues in knowledge-assisted systems; if wrong information is added to the knowledge base, assistance can lead to wrong assumptions. While preventing this is only possible on user-side, one can still try to make the system more robust against such cases, for example by sanity checks or by proposing multiple options from the knowledge base to the user such that outlier can be identified. In case of the ontology, different evaluation methods are available [159].

7.5 Conclusion

Knowledge-assisted visualization systems are playing an important role due to increasing complexity of both, data and visualization systems. With the Knowledge Rocks Framework, we support the extension of existing visualization systems to incorporate knowledge assistance.

Based on the KAVA model, we determined components that are required to make a visualization system knowledge-assisted; then, we isolated these components in several knowledge-assisted systems and derived an application-agnostic architecture to provide them. After the validation of the Knowledge Rocks Framework using the KAVA model and giving implementation pointers, we applied it to several knowledge-assisted systems to demonstrate the wide range of possible results.

As future research opportunities, more applications of our framework and possibilities to support the creation of an acting ontology on the structural and the implementation side can be investigated. Also, limitations of the Knowledge Rocks Framework with respect to the size and structure of the acting ontology can be researched. In addition, possibilities to allow editing of the acting ontology by users, guided by ontology validation and possibly frameworks that support the creation of callback functions is promising in many applications and could be supported by the framework.

Depending on the implementation of the callback functions, the classification via the acting ontology bases on causality or correlation; this can even differ between

classes. Support to distinguish these in the Knowledge Rocks Framework are an interesting research topic for the future. Also handling misleading information that got into the knowledge base is an interesting research topic for KAVA systems in general, the Knowledge Rocks Framework, and concrete implementations.

The Knowledge Assisted Security in Process System

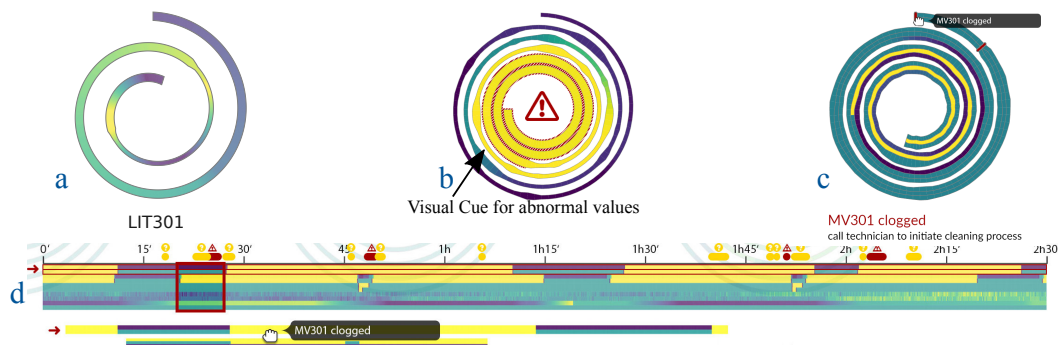


Fig. 8.1.: The **knowledge-assisted Security in Process System** bases on the Knowledge Rocks Framework. **a** Readings without detected anomaly. In case of anomalies, visual cues for abnormal values **b** and instances from the knowledge base are added to the spiral plot of the analyzed data using a stream graph **c**. The time slider is augmented to show related instances from the knowledge base **d**.

Together with Dr. Simon D. Duque Anton I applied the Knowledge Rocks Framework to the Security in Process System presented in Chapter 6. This application was published as detailed user study in the Knowledge Rocks paper at IEEE Vis 2021 [*115]. As a short recall, the requirements for the system design were:

- R1 System monitoring and triage analysis should be supported simultaneously.
- R2 Detected anomalies should be clearly highlighted in the data.
- R3 Classification of values in category II as abnormal or normal,
- R4 identification of false positives should be possible.
- R5 The displayed information and the interaction possibilities should allow identification of false negatives.
- R6 The visualization system should render triage analysis by experts as well as by laymen (in terms of cyber security) possible.

Knowledge assistance in the enhanced Security in Process System aims on additional support for

- R3 by providing the classification of similar cases from the knowledge base, allowing **comparison** and **combination** with previous results,

- R4 by **comparing** detected anomalies to false positives in the knowledge base to **separate** them from correct alerts, and overall,
- R6 by allowing collaboration and knowledge transfer between users.

In addition, enhancements of the system that are independent of the knowledge base support ■R5 by giving a detailed overview of the complete data set in the time slider.

In this chapter, I follow the implementation steps of the Knowledge Rocks Framework, outlined in Section 7.3: first, I define the knowledge base in Section 8.1. Then, I define structure and classes of the acting ontology and implement callback functions that allow an automatic traversal (Section 8.1.1). After the definition and implementation of the knowledge base's input and (automated) output in Sections 8.1.2 and 8.2, I describe how to integrate the knowledge in the Security in Process System in Section 8.3. I provide an expert evaluation and usage scenarios in Section 8.6 and discuss the results. In Chapter 9, I present user studies for the original Security in Process System and our enhancement.

8.1 Knowledge Base

The Security in Process System is used by multiple analysts concurrently and/or asynchronous in a shift work schedule. Building a knowledge base can significantly improve the usefulness of the system, providing support for understanding and decisions, especially since the system addresses both, experts and laymen (■R6).

As an anomaly detection system, the Security in Process System focuses on incidents that are defined as periods of the analyzed time series with a high abnormality rating. Shared knowledge can for example incorporate exemplary instances of proven attacks, instances of falsely detected anomalies, visualization settings for individual devices, and visual cues for specific classes of incidents. Thus, the active knowledge stored in the acting ontology is based on incidents, and it is used to classify readings of devices with high abnormality ratings. It is active in the sense that stored incidents from the database are described and classified by the knowledge in the acting ontology.

The instances stored in the database represent the passive knowledge. They consist of readings of possibly multiple devices within a fixed time frame and represent interesting values and patterns of different devices regarding one specific incident. Since different devices often correspond to different incident classes during a single

incident, an instance in the database is identified by a set of classification labels. Additional information, like for example periods of the spiral plots and chosen color maps can be stored with a link to instances and individual devices. Based on this information, guidance is provided. The database schema is shown in Figure 8.2.

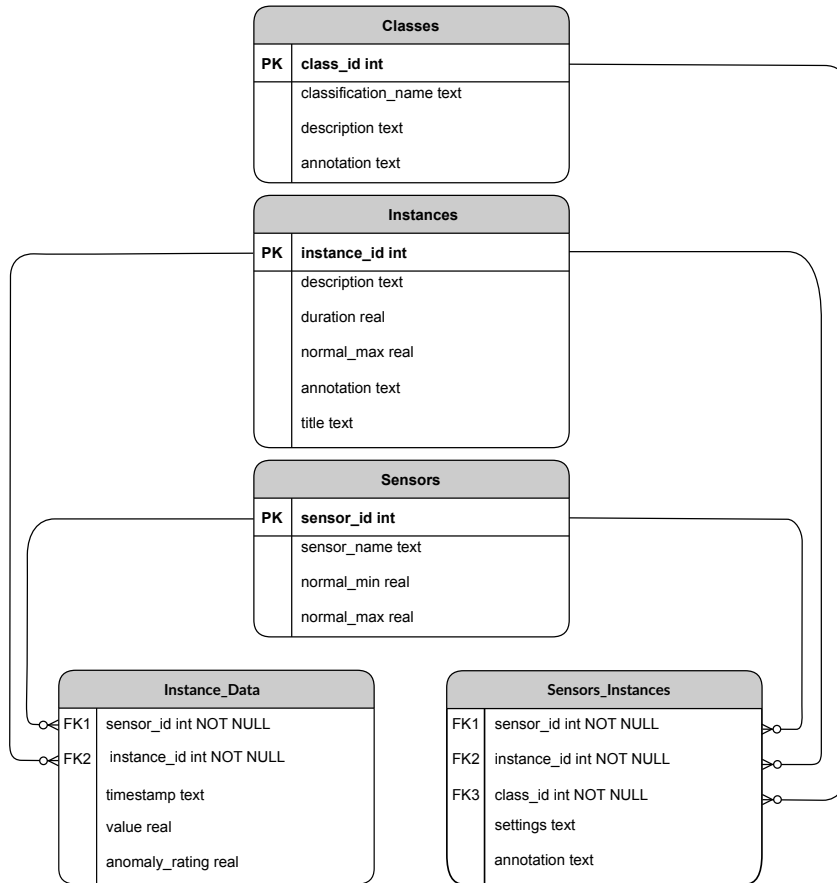


Fig. 8.2.: Database schema designed as part of the implementation of the Knowledge Rocks Framework based on the Security in Process visualization System.

8.1.1 Acting Ontology

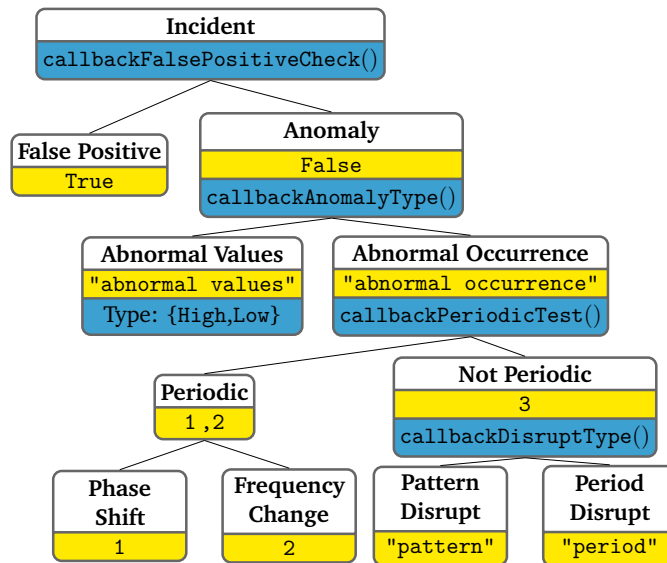


Fig. 8.3.: The **incident based acting ontology** for the Security in Process System: callback functions (blue) and properties (yellow) that are required for the traversal. The OWL code for this ontology is given in Figure 8.4 on page 147.

To classify incidents, we define the acting ontology as shown in Figure 8.3 with the following classes:

Incident. A series of readings that have been assigned a high abnormality rating; either an anomaly or false alert.

False Positive and Anomaly. As direct children of the incident class, these classes represent a false alert and the parent class for all anomaly types. An anomaly is either the occurrence of abnormal high or low values or the occurrence of values within the normal range at an unexpected time step ("Abnormal Occurrence").

Abnormal Values and Abnormal Occurrence. Normal ranges for the individual devices can be determined using readings during incident-free operation. They are stored as sensor properties in the database. Anomalies are classified as abnormal values if they contain readings outside the normal range; if applicable, they are assigned the type *High* or *Low*. Abnormal occurrence of values refers to values that are within the normal range but do not follow the normal pattern of the considered time series (for example they disrupt an established period). Depending on the

behavior of the system before and after the incident, we distinguish between periodic and not periodic abnormal occurrences.

Periodic and Not Periodic. An abnormal occurrence of values is periodic if the considered time series is periodic before and after the incident with a possible transition at a constant value between two periods. The "Not Periodic" class describes all remaining abnormal occurrences of values.

Phase Shift and Frequency Change. A periodic abnormal occurrence is a phase shift if the phase of the period is shifted and the frequency remains the same. If the frequency changes, it is a frequency change.

Pattern and Period Disrupt. A non-periodic abnormal occurrence is a period disrupt if an existing period is disrupted and does not re-occur. If there was no existing period, it is a pattern disrupt.

Note that to keep Figure 8.3 on the preceding page comprehensible, additional properties are omitted. Especially for the anomaly instances (e.g. phase shift, frequency change, pattern disrupt and period disrupt), class-based remarks, actions to be taken and visualization settings such as visual cues can be added.


```

1 <rdf:RDF>
2 <owl:Ontology rdf:about="http://www.co-ode.org/ontologies/ont.owl"/>
3 <owl:Class rdf:about="#Abnormal_High">
4 <rdfs:subClassOf rdf:resource="#Abnormal_Values"/>
5 <rdfs:isDefinedBy>5</rdfs:isDefinedBy>
6 </owl:Class>
7 <owl:Class rdf:about="#Abnormal_Low">
8 <rdfs:subClassOf rdf:resource="#Abnormal_Values"/>
9 <rdfs:isDefinedBy>6</rdfs:isDefinedBy>
10 </owl:Class>
11 <owl:Class rdf:about="#Abnormal_Occurrence">
12 <rdfs:subClassOf rdf:resource="#Anomaly"/>
13 <rdfs:comment>callback_periodic_test</rdfs:comment>
14 <rdfs:isDefinedBy>abnormal_occurrence</rdfs:isDefinedBy>
15 </owl:Class>
16 <owl:Class rdf:about="#Abnormal_Values">
17 <rdfs:subClassOf rdf:resource="#Anomaly"/>
18 <rdfs:isDefinedBy>abnormal_values</rdfs:isDefinedBy>
19 <rdfs:comment>Type: High/Low</rdfs:comment>
20 </owl:Class>
21 <owl:Class rdf:about="#Anomaly">
22 <rdfs:subClassOf rdf:resource="#Incident"/>
23 <rdfs:comment>callback_anomaly_type</rdfs:comment>
24 <rdfs:isDefinedBy>False</rdfs:isDefinedBy>
25 </owl:Class>
26 <owl:Class rdf:about="#False_Positive">
27 <rdfs:subClassOf rdf:resource="#Incident"/>
28 <rdfs:isDefinedBy>True</rdfs:isDefinedBy>
29 </owl:Class>
30 <owl:Class rdf:about="#Frequency_Change">
31 <rdfs:subClassOf rdf:resource="#Periodic"/>
32 <rdfs:isDefinedBy>2</rdfs:isDefinedBy>
33 </owl:Class>
34 <owl:Class rdf:about="#Incident">
35 <rdfs:comment>callback_false_positive_check</rdfs:comment>
36 </owl:Class>
37 <owl:Class rdf:about="#Not_Periodic">
38 <rdfs:subClassOf rdf:resource="#Abnormal_Occurrence"/>
39 <rdfs:comment>callback_disrupt_type</rdfs:comment>
40 <rdfs:isDefinedBy>3</rdfs:isDefinedBy>
41 </owl:Class>
42 <owl:Class rdf:about="#Pattern_Disrupt">
43 <rdfs:subClassOf rdf:resource="#Not_Periodic"/>
44 <rdfs:isDefinedBy rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">pattern</rdfs:
isDefinedBy>
45 </owl:Class>
46
47 <owl:Class rdf:about="#Period_Disrupt">
48 <rdfs:subClassOf rdf:resource="#Not_Periodic"/>
49 <rdfs:isDefinedBy>period</rdfs:isDefinedBy>
50 </owl:Class>
51
52 <owl:Class rdf:about="#Periodic">
53 <rdfs:subClassOf rdf:resource="#Abnormal_Occurrence"/>
54 <rdfs:isDefinedBy>1,2</rdfs:isDefinedBy>
55 </owl:Class>
56 <owl:Class rdf:about="#Phase_Shift">
57 <rdfs:subClassOf rdf:resource="#Periodic"/>
58 <rdfs:isDefinedBy>1</rdfs:isDefinedBy>
59 </owl:Class>
60 </rdf:RDF>

```

Fig. 8.4.: The OWL code for the incident based acting ontology in Figure 8.3 on page 145.

Our implementation of the callback functions includes pattern matching, machine learning components, and procedural reasoning to illustrate the variability of the Knowledge Rocks Framework.

callbackFalsePositiveCheck decides if an incident is a false positive. This can only be decided by experts concerning individual instances. Hence, this callback function is implemented as a pattern matching for existing instances in the database of type “False Positive” with a small threshold. False positives can hence only be detected if a similar example is already contained in the database. In detail, the matrix profile method by Chin-Chia et al. was used [217]. It returns `True` if the incident was recognized as a false positive and else `False`. These return values are linked to the classes “False Positive” and “Anomaly” via their properties.

The matrix profile method provides the minimum distance between two time series when moving the first series over the second. It also provides the offset in time for one of the time series that results in this minimum distance. Hence, the results of the matrix profile method are not only used to determine the similarity to existing instances of false positives in the database; they are further used to determine the optimal position (offset) for stored instances from the database with respect to the analyzed time series. That is, the method determines at which position (in time) stored instances fit best to the given data.

callbackAnomalyType determines the high-level anomaly type. Since determining whether or not values are in their normal range is straight forward, this callback function tests for abnormal values. If there are any, `abnormal values` is returned and the incident is classified with the appropriate type if applicable. Else, `abnormal occurrence` is returned and the traversal proceeds with the class “Abnormal Occurrence”. The return values can be chosen arbitrarily, they just need to be stored in the properties of the classes accordingly.

callbackPeriodicTest decides whether or not periodicity is present before and after the anomaly. This is done by testing if the anomaly can be classified as a child of the periodic class (that is as phase shift or frequency change) using two isolation forests trained for novelty detection. If one of the forests recognizes the given time series as known, the according value (1 for phase shift, 2 for frequency change) is returned. If both forests recognize the data, the value with the highest probability is returned. If both forests classify the data as novelty, 3 for “Not Periodic” is returned. Since this approach classifies more in depth than the next level in the ontology, the

class “Periodic” has a list of possible return values as property and does not have a callback function attached.

Isolation forests, as used in this callback function, are one-class classifiers. In contrast to binary or multi-class classifiers, they are trained on one class of events, which is used to generate the classification model. After this, in the testing or operating phase, new events are classified to either match or not match the classification model. Hence, their application in the acting ontology is straight forward for the binary decision whether or not the input data belongs to a certain class. This approach is often used since novelties or outliers are by default sparse and occur significantly less frequent than normal events. While common classifiers assume an even distribution of classes, one-class classifiers handle the sparsity of outliers well.

Dynamic environments often show a drift of events, meaning that over time, the values and frequencies change. If a trained classification model is used to detect outliers, the performance degrades as a consequence of this value drift. This degradation of performance can be circumvented by re-training a classification model. This approach goes well together with the mutant nature of the acting ontology.

Since the occurrence of specific classes of anomalies in the data set are limited, we used synthetic training data. Examples for the training data and the generation approach are given in Figure 8.5 on the following page. Exploiting explicit domain knowledge to improve results of machine learning processes is an active field of research [77, 199]. Machine learning techniques like isolation forests enable the acting ontology to act on the knowledge base via simulated cognitive processing [41]. Chen et al. describe knowledge-assisted visualization with simulated cognitive processing as an opportunity to overcome the shortcomings of knowledge-assisted visualization with acquired knowledge representations.

callbackDisruptType decides whether previous to the disruption a periodical pattern was present in the data. There are several ways to detect periodical behavior. We used auto-correlation as an input for a one-class classifier returning `pattern` if no period could be found and `period disrupt` if it recognized the input as periodic previous to the incident.

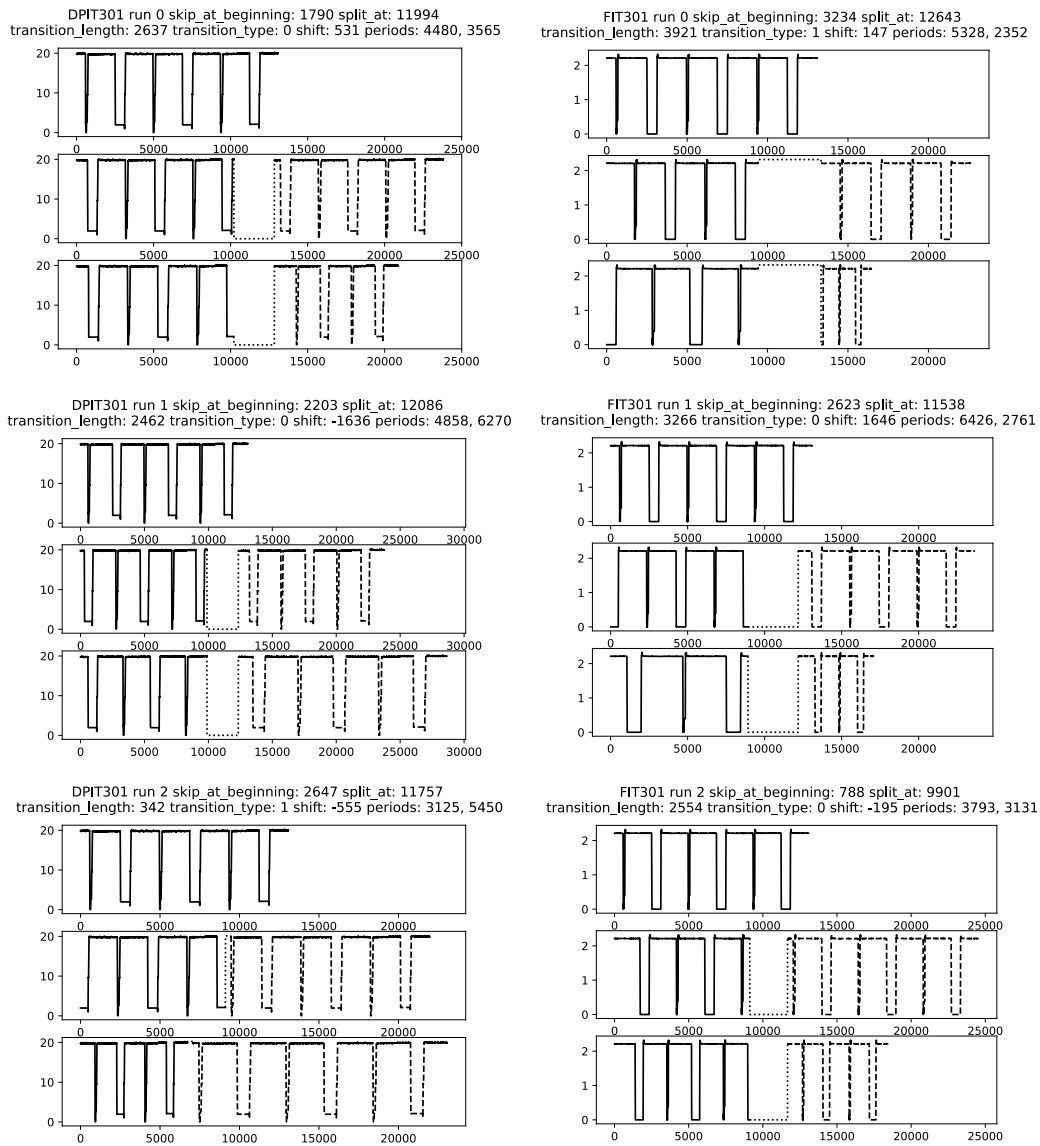


Fig. 8.5.: Training data examples for phase shift and frequency change detection. The first row of each plot contains the original device readings that were used to create training data. The following rows contain generated data with the settings provided in the plot header.

8.1.2 Suggesting Related Instances

After classification of incidents in the analyzed data using the acting ontology, related instances from the database are automatically presented to users to **compare** them as decision support (■R3, ■R4, ■R6). The selection process works as follows: the classification of incidents results in the classification label set for all devices reporting an incident $Cl_{cur} = \{(c, cl) | c \text{ device}, cl \text{ classification}\}$. In addition, the matrix profiles for every instance in the database and analyzed data have been calculated by `callbackFalsePositiveCheck`. For each device c of each stored instance \mathcal{I} , the best matching position is stored together with the calculated distances $d_{min}(c, \mathcal{I})$. Based on the set of classification labels and the best matching position for every case, the instances are ranked:

$$rank(\mathcal{I}) = \sum_{(c, cl) \in Cl_{cur} \cap Cl_{db}(\mathcal{I})} d_{min}(c, \mathcal{I}).$$

With $Cl_{db}(\mathcal{I})$ being the set of classification labels of the currently ranked stored instance \mathcal{I} . The five instances with the highest ranking are then suggested in the enhanced time slider.

8.2 Storing Instances

Figure 8.6 on the next page shows possible workflows in the enhanced Security in Process System. In case an analyst wants to store a new instance to the database, that was **separated** from existing ones, the storing mode can be entered by pressing a GUI button. The devices contained in the instance can be selected via mouse click. Their classification by the acting ontology is presented and users can change it using a drop down list containing all classes from the acting ontology. To get support for the classification, users can access the ontology visualization with the given class hierarchy and annotations from the callback functions.

In addition to the classification, annotations can be added for each device and for the whole instance. After entering a name for the instance, it is saved with a second click on the storing button. The stored properties are the selected devices and time frame with the corresponding data and anomaly ratings, annotations, and current visualization properties: period, selected color map and color map reference. After instances are annotated by experts and added to the knowledge base, laymen are able to benefit from the stored knowledge (■R6).

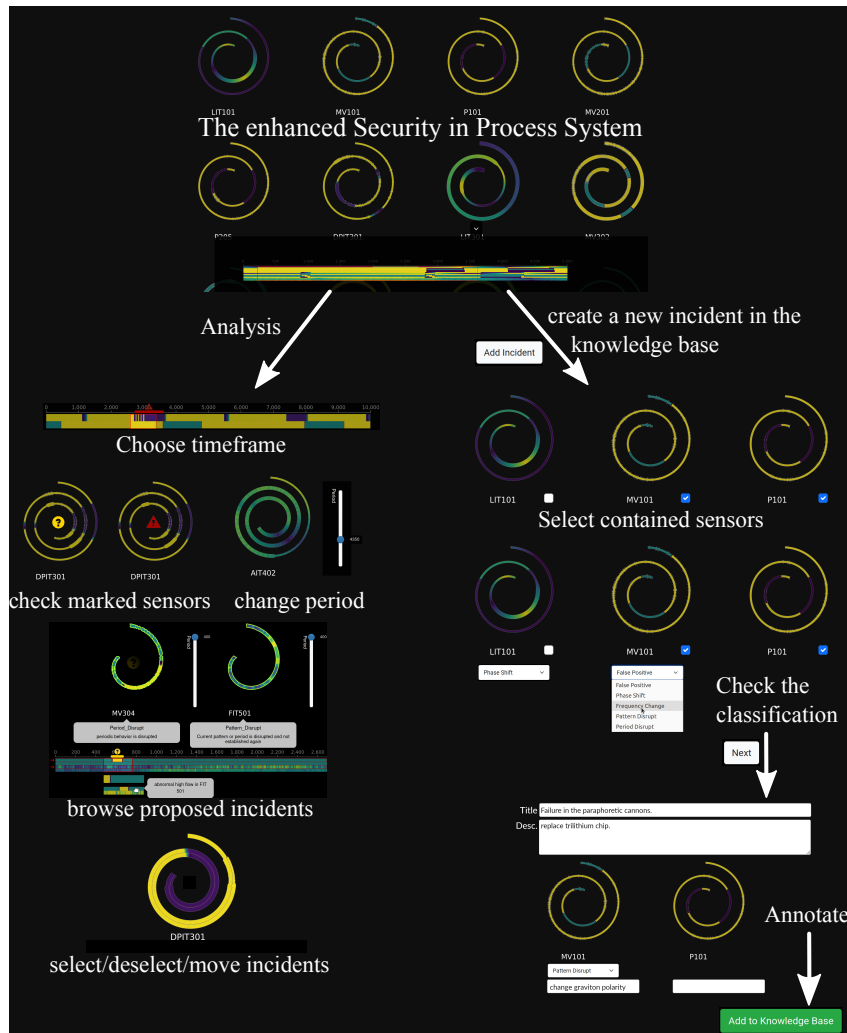


Fig. 8.6.: Workflow in the enhanced Security in Process System.

8.3 Visualization

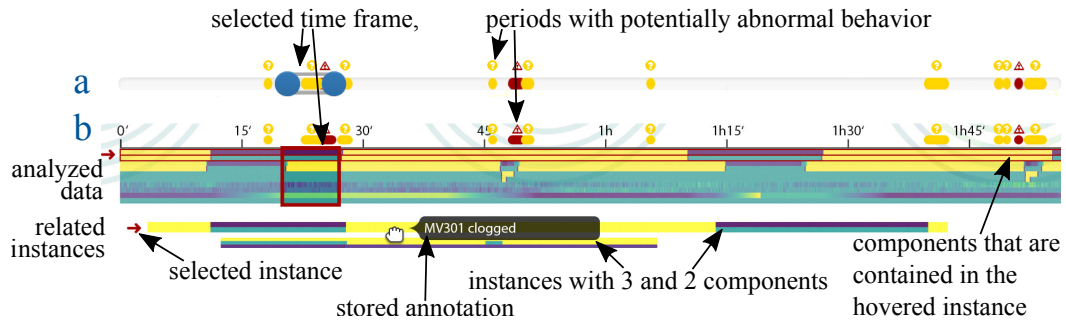


Fig. 8.7.: Original **a** and enhanced time slider **b**: The line graph now gives an overview of the readings of all devices and related instances from the knowledge base. Clustering ensures that devices with similar readings are adjacent.

As described in Section 7.2.2 on page 130, the visualization of the Security in Process System needs to be extended to incorporate knowledge assistance. In contrast to the original system, we implemented the extended Security in Process System using D3.js [24] and python 3, framed by bottle [81].

The Enhanced Time Slider. The time slider now includes information on the sensor readings and presents related instances from the knowledge base (Figure 8.7). To render this possible, we used a line graph following ideas by Kincaid et al. [103]. The analyzed data is shown on top, in a strand of device readings. Potentially abnormal behavior is highlighted as in the previous version with symbols and colored bars above the slider. Below the analyzed data, related instances from the database are suggested with the same visual encoding and ordering to support **comparison** and **combination**. Using the mouse, they can be selected, de-selected, and moved relative to the analyzed data. The initial position of an instance is chosen according to the minimum distance as described in Section 8.1.2 on page 151. When an instance is hovered with the mouse, contained devices are highlighted in the time slider and the spiral chart; in addition, stored annotations for the hovered instance are shown as tool tip.

We optionally cluster the devices' readings to ensure that devices with similar patterns in their readings are adjacent in the time slider, supporting the **comparison** and **combination** of devices with similar behavior. The spiral plots in the spiral chart (b in Figure 6.3 on page 108) are re-ordered accordingly. Forming flat clusters is achieved using the inconsistency method in SciPy [197]. Similar to the previous version of the time slider, the selection frame can be re-positioned via drag and drop and its width can be changed with handles on the borders.

The enhancements of the time slider give an overview of the complete data set that was missing in the original Security in Process System. Providing this overview supports the identification of areas with abnormal behavior, even if the anomaly rating is low (R5) and supports **comparison** of readings across the whole data set.

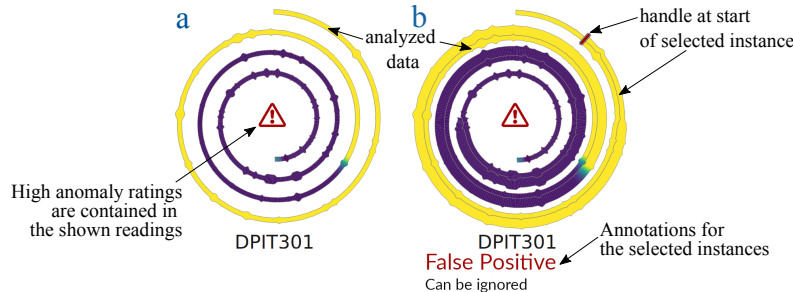


Fig. 8.8.: Original **a** and enhanced spiral plot **b**: Readings can be compared and annotations provide decision support.

The Enhanced Spiral Plot. Selected instances are shown in the spiral chart: we extend the spiral plot by combining it with a stream graph similar to the approach by Jiang et al. [95] (c in Figure 8.1 on page 142 and Figure 8.8). Using a stream graph that is centered at the spiral's center line, the thickness of the analyzed data and **compared** and **combined** instances accumulates. The analyzed data is always the innermost spiral. Handles at the beginning of a selected instance allow moving the instance with respect to the analyzed data; this is in addition to dragging the instance under the line graph. Annotations of the instance or its devices are shown on top of the spiral chart or below the according spiral plots respectively. Since not only the sequence of events but also their duration is important when searching for anomalies in industrial process data, selected instances are always shown with the same period as the analyzed data. Otherwise it would for example be possible to hide or create differences in the period.

Since spacing between spiral twists is an issue if multiple instances are selected or the anomaly rating is high, the maximum thickness for each strand is limited. If necessary, this limit is automatically reduced. In addition, it would be possible to change the selected period of the spiral to one of its integral multiples. Doing so, less twists are rendered but patterns are still recognizable.

Ontology Visualization. Direct access to the knowledge base is only required to support the manual classification of instances. To implement this, we implement a basic ontology visualization in a browsable tree layout similar to Figure 8.3 on

page 145. Having access to the different classes of the ontology supports users (especially laymen) in collaborating and identifying anomalies (■R3, ■R4, ■R6). This is because having a taxonomy and dependencies between classes provides a framework to classify incidents and thus rate them appropriately.

8.4 Guidance

Proposed incidents below the time slider can be chosen freely by the user, providing annotated examples to compare the current data with. Prescribing specifications can optionally be invoked by pressing the Alt-Key when selecting a related instance below the time slider. Then, linked visualization settings from the database are prescribed. A classification of an incident as “Abnormal Values” triggers highlighting of the period containing the abnormal values (b in Figure 8.1 on page 142), supporting ■R3 and ■R6. Additional visual cues supporting the identification of patterns could be implemented following the ideas by Ceneda et al. for spiral plots [38].

The provided guidance is an optional addition to the Security in Process System. If none of the proposed incidents fits a situation where a user requires guidance (or there is no incident proposed), the browsable ontology and automated classification still provide support. In addition, users are urged to add the incident to the knowledge base in such a case.

8.5 Data Size and Storage Access

The considered data set consists of readings and anomaly ratings for 28 sensors at 467,919 time points. To create the enhanced time slider, every time step of all selected sensors needs to be loaded from the server. If static data is analyzed, this means that there is some seconds of loading time at system start. To keep this loading times small, the resolution of the transferred readings outside the selected time frame can be reduced. Changing the selected time frame, high resolution data for this frame is loaded dynamically. Streaming data, only added time steps need to be added to the right of the slider and cropped from the left, allowing fluent streaming.

The knowledge base we created during the creation of the system consists of 26 incidents containing 23 different sensors over 615,844 time steps in total. Currently, at system start all incidents are loaded to ensure fast access times. With a growing

knowledge base, more elaborate storage management is required, for example keeping incidents in memory based on their access frequency. During system use, stored incidents are accessed every time a new sensor is selected or the selected time frame is changed. Queries on the knowledge base during system use can be performed asynchronous in the background via ajax. By running a local server, access times are negligible. Remote access to the system might result in longer loading times. Classification of sensor readings via the acting ontology is done in milliseconds. Of course this runtime depends on the implementation of the callback functions.

8.6 Usage Scenarios and Expert Evaluation

I interviewed an it-security expert in the context of exemplary usage scenarios of the knowledge-assisted Security in Process System.

Abnormal High Values. In the time slider in Figure 8.9 an anomaly is indicated around 3h 10'. While the frequency change in the readings of several devices is quite obvious (**combine**), the sensor at the bottom of the time slider faces an incident classified as “Abnormal Values” of type “High”. The linked visual cue highlights and **separates** the period with abnormal high values in the spiral plot (b in Figure 8.1 on page 142). While the frequency changes are much more striking than the high values, the recorded attack actually originates on this device. The sensor was attacked, and the value was raised to provoke the recorded reaction in different actuators.

The interviewed expert found it useful to be able to spot severe anomalies already in the time slider and independent of the results of the anomaly detection (■R5). According to him, the visual accentuation of different classified anomalies is a way to directly share knowledge with domain experts; visual cues are immediately visible and helpful (■R3). Especially values outside the normal range usually indicate severe incidents, be it a damage in the machine or an intrusion. Adding annotations

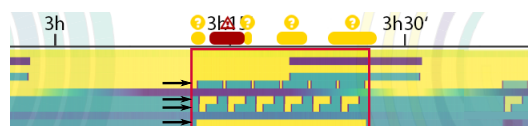


Fig. 8.9.: Overview using the time slider: Significant pattern changes can be spotted in the enhanced time slider.

to the stored incidents made sense to the expert because they enable laymen to react appropriately and support experts in troubleshooting (■R6).

False Positive Identification. In spiral a in Figure 8.8 on page 154, the high abnormality rating for sensor DPIT301 triggers an alert, but there is no anomaly visible in the readings. The decision to disagree with the anomaly detection process is now supported by a related instance classified as “False Positive”. Matching the analyzed data with the stored incidents in the database results in a very close match with this instance. Thus, it is suggested as related and selecting it, the incident is added to the spiral plot at the calculated optimal position. **Comparing** the readings, the similarity becomes clearly visible (Figure 8.8b). With this support, we expect it to be easier to identify false alerts by the anomaly detection system (■R4) and **separate** them from actual incidents.

The intuitive incorporation of this very specific expert knowledge appeared fruitful to the interviewed expert. In his opinion, expanding the database with both, common and specific cases, to form a big knowledge base will be a huge support in triage analysis.

Creating and Using the Acting Ontology. The interviewed expert was enthusiastic to build a crucial part of the extended system as an ontology. In general, he finds it helpful to categorize and classify incidents, and comparatively easy to build the ontology. According to him, conducting research regarding the cause of an anomaly is facilitated by its classification. In his opinion, also the externalized expert knowledge on different anomaly types is helpful to get an overview, an idea of anomaly sources, and to manually classify anomalies.

The expert highlighted positively that the ontology inherently introduces fundamental concepts that facilitate communication between experts and laymen using the system (■R6). Also, he saw great potential in the flexibility of the Knowledge Rocks Framework architecture, allowing for example one-class classifiers in the callback functions.

Further Development. As possible further development of the system, the expert recommended an automated prompt to add instances to the knowledge base where closely related instances are missing in the knowledge base, and to make the knowledge base (that is ontology and stored instances) accessible independently of the system for training purposes.

8.7 Conclusion and Future Work

The Knowledge Rocks Framework proved successful in its application to the Security in Process System. By following the required implementation steps, we added knowledge support, allowing collaboration and providing assistance in crucial and basic analysis tasks.

There are several opportunities for further enhancements of the system. To simplify access to the stored knowledge, additional visual cues for more anomaly classes are an interesting option. Also, making the knowledge base accessible independently of the system for training purposes is an additional option to benefit from the stored knowledge.

To support the growth of the knowledge base, an automated prompt to add instances to the knowledge base where closely related instances are missing can be helpful. Also, the possibility to add unknown incidents with a “request” for annotations, that are then classified and annotated by experts will improve the knowledge base and boost collaboration between experts and laymen. Finally, mechanisms to prevent misleading knowledge in the knowledge base need to be researched and implemented.

We evaluated the Security in Process System in its original and enhanced version in user studies and give the results in the next Chapter 9.

Evaluation of the Security in Process System

To evaluate our design of the Security in Process System (the *original system*) presented in Chapter 6 and of the knowledge-assisted, enhanced Security in Process System (the *enhanced system*) presented in Chapter 8, I performed two detailed user studies in cooperation with the HCI Group at the Technische Universität Kaiserslautern.

While the first evaluation was part of the Security in Process paper [*116], the evaluation of the knowledge assisted Security in Process System was published stand alone as a short paper at the TrEx Workshop taking place at IEEE Vis 2021 [*119] in collaboration with Vera Memmesheimer, Frederike Gartzky and Christoph Garth. Together with Vera Memmesheimer of the HCI Group, I developed the questionnaire which was then implemented in an online tool by Frederike Gartzky. The results were evaluated and interpreted by Vera Memmesheimer with my support.

The overall result of both evaluations show that triage analysis using the Security in Process System is effective, comfortable and superior to naïve time series visualizations. In both systems, a learning effect in terms of task completion time for correct responses was observable. Using the enhanced system, more anomalies and false positives were identified correctly. Overall, the Security in Process System with assistance turned out to be the preferred system.

After giving details on the experimental design, performed tasks and the study set up in Sections 9.1, 9.2 and 9.3, I present the the study results concerning effectiveness, satisfaction, efficiency and cognitive load in Section 9.4 and discuss them in Section 9.5.

9.1 Experimental Design and Procedure

To assess requirement ■R6, the ability of laymen (in terms of cyber security) to verify results of the anomaly detection using our system was evaluated by 15 participants without background in cyber security in both studies.

In the evaluation of the original system, 11 participants had a technical background in IT or electrical engineering. After a short introduction to the system, the users performed several tasks (see Section 9.2) and filled the questionnaire on effectiveness along the way. Afterwards, they were asked to fill a questionnaire on usability of the system.

The participants evaluating the enhanced system consisted of 6 women and 9 men aged 24 - 44 where 13 participants had a technical background. Furthermore, some participants had experience in visual analytics (7), with spiral plots (5) and with the original system (3). A within-subject design was used to compare the usability of the original and the enhanced system: the participants were asked to complete several tasks (see Section 9.2), either once with the original and once with the enhanced system, or only with the enhanced system if the task related to a novelty without counterpart in the original system. To avoid learning effects, the order of the systems was assigned randomly. In total, 9 participants started with the original and 6 with the enhanced system. Because of the Covid-19 pandemic, this evaluation was executed online with the evaluation tool presented in Section 9.3 on page 163. Prior to each task, we provided an explanatory video to introduce the system and establish a common knowledge basis. Each video had to be watched at least once and could be replayed throughout the completion of the task.

In line with ISO 9241 - 10 and 11, we assessed the system usability for both systems. For the enhanced system, we measured success rates and task completion times throughout the experiment. Furthermore, we posed questionnaires about satisfaction and cognitive load after the completion of the tasks with original and enhanced system respectively. At the end of the experiment, we asked for the preferred system.

9.2 Tasks

Overall, 7 different tasks were completed during the evaluations. The evaluation of the original system was conducted to investigate the usability and the fulfillment of the defined system requirements. The evaluation of the enhanced system on the other hand does not consider all features that are present in the original system, but focused on the comparison of original and enhanced system, and the impact of knowledge assistance. Hence, some tasks were only completed during the evaluation of the original system, and some tasks are only applicable to the enhanced system.



Fig. 9.1.: The original time slider with warning and alert labels.

▲ **T0 Thread Identification.** The participants determined if the system is currently under attack, and the time point where a thread occurs for the first time in the considered data set. This task was only performed in the evaluation of the original system, based on the warning ⚠️ and alert 🚨 labels in the original time slider (Figure 9.1).

▲ **T1 Risk Evaluation.** In this task, we presented readings rated as warnings by the automated anomaly detection. The participants were asked to decide whether the readings are abnormal or normal. For the evaluation of the original system, one warning that was an actual anomaly and one warning on readings with normal behavior was presented. Evaluating the enhanced system, the participants considered three cases of abnormal behavior and one with normal behavior. In the enhanced system, proposed incidents from the knowledge base supported the user’s decision similar to Figure 9.2.

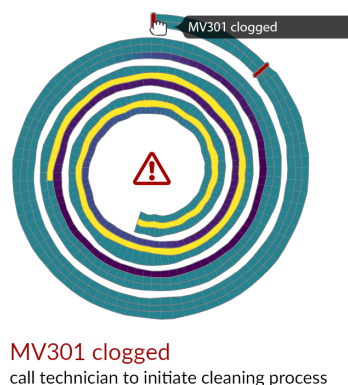


Fig. 9.2.: This spiral plot contains **readings rated as alert**. Hovering or selecting a suggested instance from the knowledge base, contained devices are highlighted in the time slider and the spiral chart. Annotations are shown for contained devices and the instance.

▲ **T2 Alert Revision.** The participants were asked to check readings that were rated as attacks for abnormal behavior, and identify each example as true or false positive. In the evaluation of the original system, each case was considered once. Evaluating the enhanced system, two true positives and two false positives were identified by the participants using once the original and once the enhanced system. Again,

stored incidents from the database could be selected using the enhanced system, supporting the decision (Figure 9.2 on the preceding page).

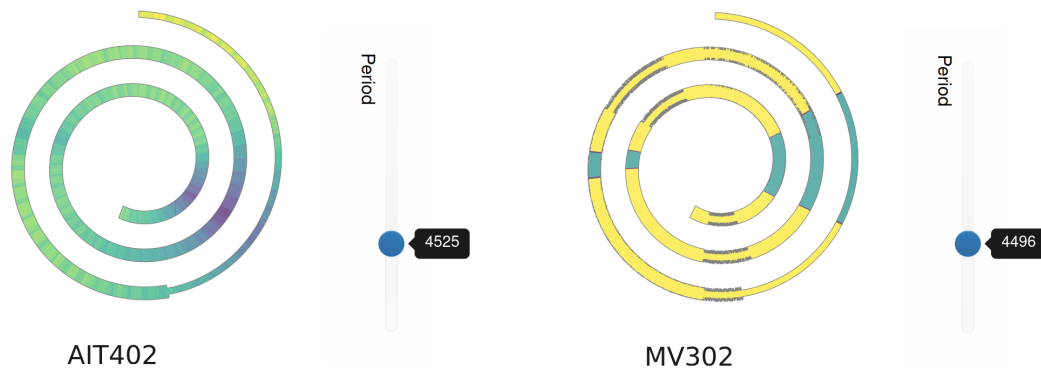


Fig. 9.3.: Periods with one (left) and two peaks (right).

▲T3 Determining Periods. The participants were asked to use the period slider to find the period of the shown sensor readings. In the evaluation of the original system, one period with one peak and one with two peaks had to be determined (Figure 9.3). Evaluating the enhanced system, two examples with two peaks per period and one example with one peak were treated using the original and the enhanced system. In the enhanced system, participants could apply stored periods from the knowledge base to the spiral plots.

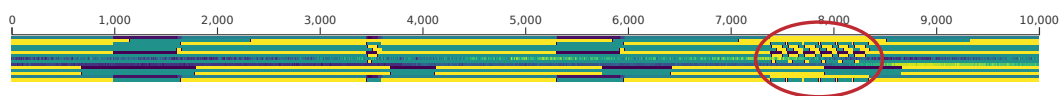


Fig. 9.4.: Irregularities in the enhanced time slider.

▲T4 Spotting Irregularities. The original and the enhanced system provide different support to spot suspicious measurements in an area detected as normal. In the original system, participants were directed to a time frame with an undetected anomaly and decided based on the spiral plots whether there is a false negative in the anomaly detection or not. Furthermore, they were asked to determine the device with abnormal behavior to capture their reasoning. In the enhanced system, the enhanced time slider provides an overview of the complete data, such that a direction to a specific time frame is not necessary any more. To evaluate this, we presented a time slider containing an anomaly (Figure 9.4) and asked the participants to enter the start time of a suspicious pattern.

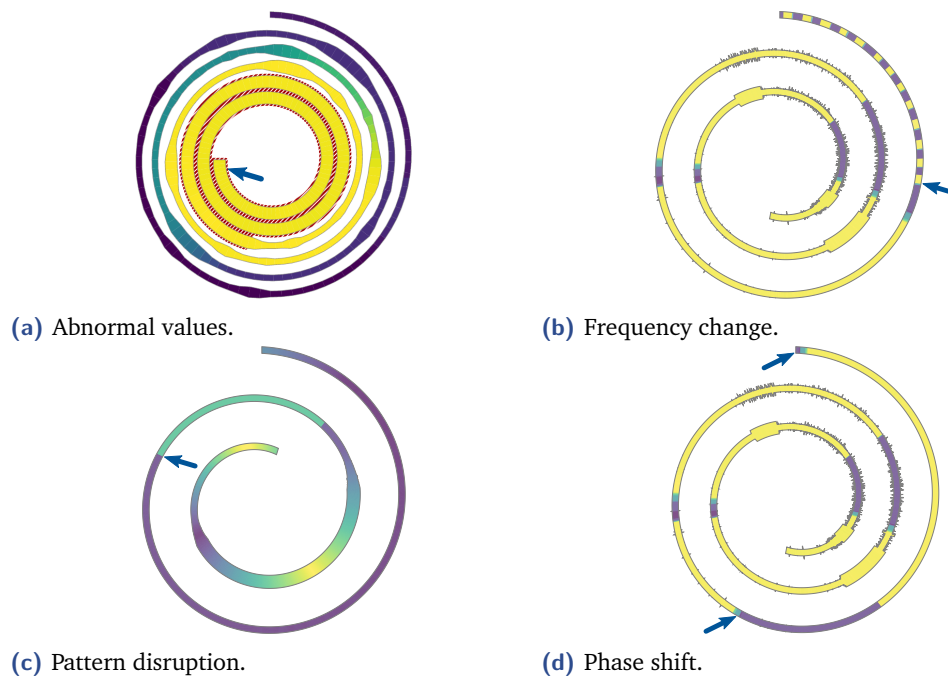


Fig. 9.5.: Examples for **different incident types**.

▲ **T5 Incident Classification.** In this task, incidents in the sensor readings had to be classified according to the ontology part of the knowledge base in the enhanced system. To this end, we provided four out of five spiral plots representing a pattern disrupt, period disrupt, phase shift, frequency shift, and abnormal high values along with a picture of the ontology and background information on frequency and phase of signals. See Figure 9.5 for the used examples.

▲ **T6 Determining Affected Devices.** To evaluate the warning and alert indications in the spiral plots, participants of the evaluation of the original system were asked to determine devices that are affected by a specific anomaly.

Table 9.1 on the following page gives an overview of the requirements that are evaluated with the different tasks and in which evaluation the tasks were executed.

9.3 Evaluation Setup

Evaluating the Original System. All participants used the same workstation. After a short introduction by me, based on a hand out with explanations of the system, they went through a questionnaire that guided them through the different tasks.

Task	Evaluated	System
▲T0	■R1 Support of monitoring and analysis, ■R2 Highlight anomalies	original
▲T1	■R3 Classification of warnings	both
▲T2	■R4 Identification of false positives	both
▲T3	■R1 Support of monitoring and analysis	both
▲T4	■R5 Identification of false negatives	both
▲T5	■R6 Support for laymen and experts	enhanced
▲T6	■R2 Highlight anomalies	original

Tab. 9.1.: Evaluated requirements and the system that was evaluated with the respective task.

The answers were written directly in the questionnaire and were either yes/no (▲T0, ▲T1, ▲T2, ▲T6), or numbers (▲T3, ▲T4). In some tasks, devices that are affected by an attack needed to be identified (▲T0). There, the participants were asked to provide a list of all affected devices.

Evaluating the Enhanced System Online. To be able to carry out an evaluation of the enhanced system while respecting social distancing, the evaluation took place purely remote. Participants independently navigated through our web-based evaluation tool that guided them through different tasks that were solved using both, the original or the enhanced system, whereby the the order of the systems was chosen randomly. In the beginning, participants were asked to give personal information and watch a video explaining the purpose of the system. Afterwards, they navigated via buttons through the different tasks and sub-tasks, where each main task was again explained by a video. Depending on the task, the participants had to provide their answer via single choice (▲T1, ▲T2, ▲T5) or via entering a number (▲T3, ▲T4). We tracked the answers and task completion times.

To ensure that all participants are able to execute all tasks although we were not able to supervise the behavior of the system, we narrowed the available functionality of the system down to specific examples. This was not the case for the evaluation of the original system where users were free to use all available features, and we were able to intervene if necessary. Because of this different setting, the focus of the evaluation of the enhanced system lies on the comparison of single features in original and enhanced system, instead of a general usability study. Furthermore, the restricted functionality lead to different ratings and user satisfaction in both user studies. Nonetheless, we are confident that the results of the evaluation of the original system concerning features that exist in both, original and enhanced system, also apply to the enhanced system.

9.4 Results

With the execution of both user studies, we gained the following results on effectiveness, satisfaction, cognitive load and efficiency.

9.4.1 Effectiveness

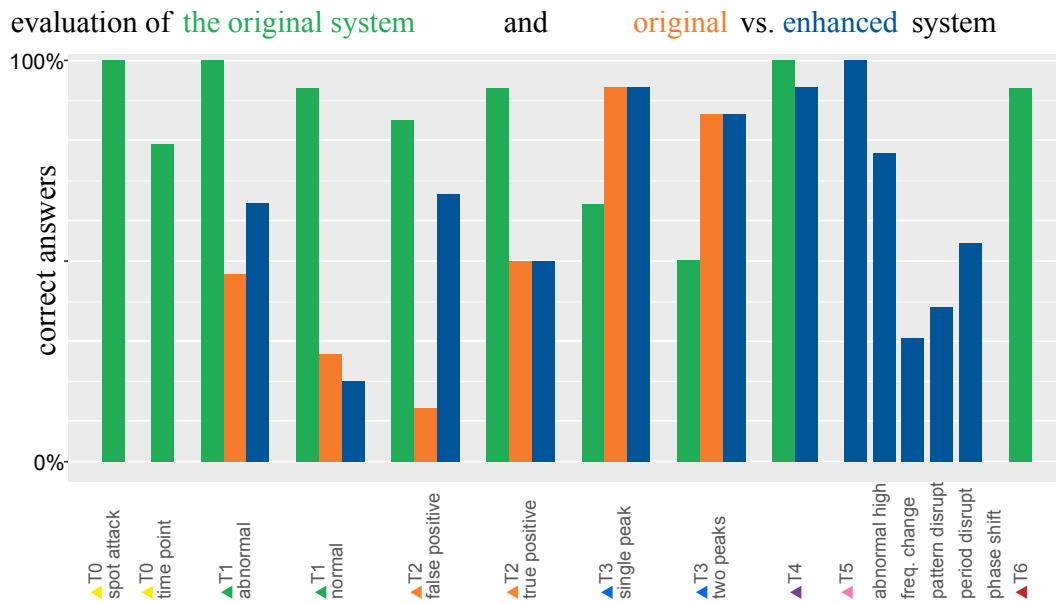


Fig. 9.6.: Evaluation results for effectiveness assessed via success rates.

We assessed effectiveness in terms of success rates. To this end, we measured whether participants selected the correct response option in tasks ▲T1, ▲T2, and ▲T5. For ▲T3 we accepted answers in a range of 500s; for ▲T4 the frame for correct responses was 960s wide. Both ranges were determined example based. The results of both user studies are given in Figure 9.6.

Effectiveness of the Original System. In the evaluation of the original system, all participants were able to identify a current potential thread in the system and 79% were able to determine correctly the first time point a thread occurs in the overall data (▲T0). The period of device data at a given time interval was determined correctly by 64% in case of a period with a single peak and by 50% in case of a period with multiple peaks per period (▲T3). While these numbers are comparably low, 93% of the participants were able to detect some kind of pattern for a chosen spiral in the spiral plot. Hence, the advantage of spiral plot visualization concerning the detection

of disrupted or shifted periods is still present, even though the correct period was not found. These results and the fact that 93% of the participants were able to correctly determine the devices that were affected by a detected anomaly (▲T6) lead us to towards opinion that requirements ■R1 (support of monitoring and analysis) and ■R2 (highlight anomalies) were implemented successfully in our original system. Considering the classification of values with warnings in abnormal and normal behavior (▲T1), 100% of all participants were able to determine abnormal behavior. The decision that an area of category II is actually normal was made correctly by 93%. With these results, we consider requirement ■R3 (classification of warnings) as met. 93% of all participants were able to identify a detected anomaly as a true positive (▲T2) and the detection of false positives was performed with 85% success rate by the participants. Hence, also requirement ■R4 (identification of false positives) is met by our original system. Requirement ■R5 (support of the system in finding false negatives) was assessed by having the participants navigate to an area marked as normal by the system that actually contained an attack. 100% of all participants were able to identify this attack (▲T4) and 93% were able to determine the device with abnormal behavior correctly. Since this question was last in our questionnaire, we see this perfect result as a hint to an existing training effect, that was already present in a 10 minute survey. Also we consider requirement ■R5 as met. Requirement ■R6, rendering verification by laymen (in terms of cyber security) possible, was obviously met since all participants were able to understand their tasks and perform most of them correctly. Overall, the participants navigated confidently through the data in our visualization system and were able to make use of the provided features to perform the tasks even after a short introduction to the system. The visualization was described as pleasant, the dark visualization mode was a clear favorite.

Effectiveness of the Enhanced System. Comparing the enhanced and the original system, higher success rates were found for the enhanced system in ▲T1 and ▲T2. As shown in Figure 9.6 on the preceding page, we found that the implemented knowledge assistance especially supports the correct detection of anomalies and false positives. Using the enhanced system, the participants identified more presented anomalies (64%) than with the original system (47%). In ▲T2, the false positive example was identified by 67% of the participants using the enhanced, but only by 13% using the original system. Concerning ▲T3, both systems performed equally well in terms of success rates. In total, 89% of the periods were determined correctly, affirming again the suitability of spiral plots for periodic behavior.

▲T4 and ▲T5 were only performed with the enhanced system. We found a particularly high success rate (93%) in ▲T4, showing that anomalies can be detected effectively with the additional information provided by the enhanced time slider. Using our ontology, 31% of the pattern and 38% of the period disruptions were classified correctly. Phase shifts (55%) and frequency changes (77%) were identified more often. 100% of the examples representing abnormal high values were classified correctly, demonstrating the high effectiveness of the visual cue in the enhanced spiral plot.

Comparing Both User Studies. Although both user studies were executed in different settings and foci, a comparison of their results provides further insights. Overall, ▲T1, ▲T2, ▲T3 and ▲T4 were performed in both user studies. To obtain conclusive results in the evaluation of the enhanced system, our choices of presented examples in ▲T1 and ▲T2 were more difficult than in the evaluation of the original system; having success rates around 100% already for the original system would not allow a significant increase for the enhanced system. Thus, the success rate of the enhanced system for the risk evaluation of abnormal values and the detection of false positives is even more encouraging. In ▲T3, the results for the enhanced and original system in the evaluation of original vs. enhanced system outperform the results for the original system. This could be caused by training effects and on the fact that in the evaluation of the enhanced system, more participants had a technical background. Although task ▲T4 was carried out in differing settings –in the original setting the participants spotted irregularities in the spiral plots for a provided time frame while in the enhanced system, only the enhanced time slider was available to spot them and no time frame was provided– the results are surprisingly similar. Hence, we think that having access not only to the enhanced time slider but also to the spiral chart, the results for the enhanced system could even improve further.

9.4.2 Satisfaction

To assess and compare the user satisfaction with the original and the enhance system, the participants filled a questionnaire in both user studies and select their preferred system. They indicated their agreement with 9 respectively 11 statements on a five-level Likert scale (predominantly disagree (1) - predominantly agree (5)) about the systems suitability for the tasks (Q1-Q5), controllability (Q6), conformity with user expectations (Q7), learnability (Q8, Q9), and overall satisfaction (Q10, Q11). The last two questions were only answered in the evaluation of the enhanced system. Answers of both evaluations are aggregated in Figure 9.7 on the next page.

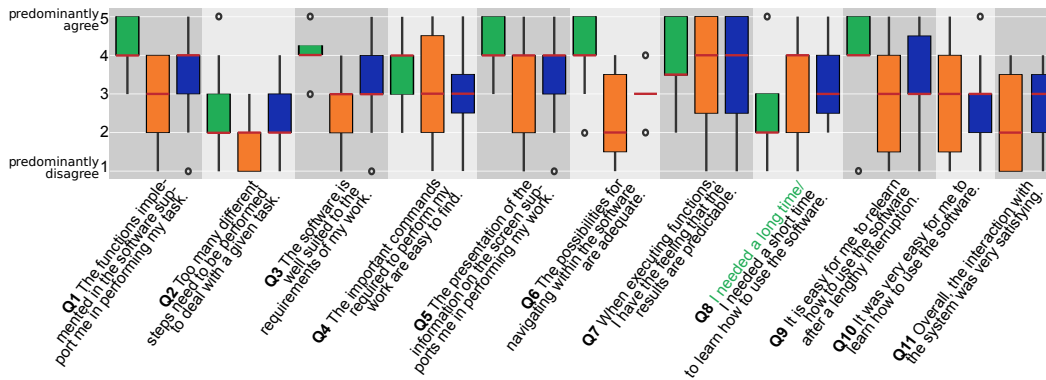


Fig. 9.7.: Evaluation results for satisfaction assessed via questionnaire.

Satisfaction with the Original System. Suitability of the visualization system for the performed tasks was evaluated as very good by the participants. In their opinion, the well-suited (Q3) and supportive commands (Q1) are relatively easy to find (Q4) and not too many steps are necessary to achieve results (Q2). Controllability was rated as very good (Q6) as well as conformity with user expectations (Q7). Several users suggested to improve the navigation in the time slider by providing a possibility to enter timestamps directly. This feature was added after the evaluation. Our previous hypothesis that handling the visualization system is easy to learn is confirmed by the participants impressions (Q8, Q9).

The effectiveness and satisfaction / usability assessment results shown are positive. We interpret these results to indicate that the metaphors of our visualization system are well chosen and that it is able to effectively support triage analysis in the industrial context.

Comparing Satisfaction with Original and Enhanced System. As presented in Figure 9.7, the functions offered by the enhanced system were rated to be more supportive (Q1) and the software to be more suited for task performance (Q3). Similarly, the navigation in the enhanced system was rated better than in the original system (Q6). Both systems received relatively low scores in (Q2), indicating that the number of steps to perform a task is adequate. We believe that the slightly higher score assigned to the enhanced system reflects the available knowledge assistance functions. Both systems received especially high scores concerning the predictability of results (Q7). It is easier for the participants to relearn the enhanced system (Q9) and the overall interaction with it was rated more satisfying (Q11). In total, 80% of the participants preferred the enhanced over the original system, affirming that knowledge assistance is indeed helpful.

Comparing Both User Studies. In the first user study evaluating the original system, the implemented functions were rated as more supportive than in the second evaluation (Q1). This reflects the restricted access to system features in the second user study – participants were not able to use the implemented functions to full capacity. The same holds true for Q3 and Q4 where the suitability of the system and the accessibility of helpful commands was rated. Also the presentation on the screen was restricted in the online evaluation, leading to lower ratings for Q5 and Q6. Especially Q6 is barely expressive for the evaluation of the enhanced system, since the navigation possibilities within the system were almost none. Most navigation took place via the evaluation website from task to task. Q8 is difficult to compare because of differently posed questions. The difference in the rating of re-learnability (Q9) could be explained by a lacking overview of the system in the second evaluation, since only limited functions in different settings were available, without connection between the examples as it was done in the evaluation of the original system.

9.4.3 Efficiency and Cognitive Load

Throughout the evaluation of the enhanced system, we recorded task completion times to assess the efficiency. We measured a median task completion time of 45s for ▲T4 and of 173s for completing the four examples in ▲T5. To compare the original and enhanced system, we considered the task completion time of the fastest correct response for ▲T1, ▲T2, and ▲T3. However, we did not find that one of the two systems generally lead to a faster completion of the tasks when comparing the respective task completion times for each participant. Instead, we found that most participants were able to complete ▲T1 (80%) and ▲T3 (93%) faster with the second system independent of the order of the systems. For these participants, we measured an average task completion time decrease of 52% in ▲T1 and of 43% in ▲T3 when comparing the fastest correct responses using the first to using the second system.

Since cognitive overload could impede the fast and correct identification of attacks during extended periods of usage, we measured cognitive load via the NASA-task load index [78] during the evaluation of the enhanced system. To compute the weighted rating (0 – 100), we followed the procedure described by the NASA [142]: the participants were asked to rate mental, physical, and temporal demand as well as performance, effort, and frustration after task completion with each system, and to weight the sources of workload at the end of the study.

While the average weighted workload rating measured for the enhanced system (65) was slightly higher compared to the original (59), we observed the overall weighted workload and ratings for the single workload scales to differ highly among participants and systems. In total, 6 out of 15 participants experienced lower workload with enhanced than with the original system.

9.5 Discussion of the Evaluation Results

Our Security in Process System exploits typical patterns that are often inherent in sensor and actuator readings from industrial processes. This enables cyber security experts and laymen to perform triage analysis and monitoring of the system simultaneously. The main characteristic of our system are manipulable spiral plots, that combine the visualization of sensor readings in their coloring with the results from anomaly detection in their line thickness. Anomalies are highlighted using further pre-attentive properties like form, movement, and dedicated colors.


We improve the support for triage analysis in OT networks in the enhanced Security in Process System. Based on the Knowledge Rocks Framework, we integrated a knowledge base; automatically suggested incidents from this knowledge base are then incorporated in the visualization, providing direct support without additional hurdles.

Based on the evaluation results, we believe our visualization systems are a useful tool to make available not only for incident response or cyber threat hunting teams working in the operational technology environment, but also for industrial companies without dedicated cyber security staff. In general, industrial control systems providing anomaly scores on periodical data could profit from our visualization approach.

In both evaluations, we found that periodicity could be monitored effectively and users were able to spot changes in the periodic behavior of a device. Enabling users to change the period of the spiral plots not only increases accuracy of the shown period but also allows the user to react to changes in the period during the process.

The evaluation with laymen w.r.t. cyber security clearly indicates that basic triage analysis tasks can indeed be performed by them using our visualization systems. Also, monitoring the readings of available devices in order to gain an overview of the current process was possible. We are convinced that staff that is trained to work with

the according machine performs even better in monitoring and detecting anomalies in the behavior and could benefit considerably from our visualization tool. Based on the learning effects we encountered in both evaluations, we also expect performance to further increase after additional training, and rate our design as appropriate for supporting triage analysis. This is also supported by the observed decrease in task completion time from the first to the second system in the evaluation of the enhanced system. We believe that in addition, appropriate training will support collaboration via the knowledge base and thus help leveraging the benefits of the knowledge assistance provided in the enhanced system.

All requirements  R1-6 were met successfully. The enhanced system added knowledge assistance and collaboration possibilities. The majority of participants selected the enhanced system as preferred and rated it to be more satisfying to interact with, easier to relearn, and its additional functions to be more supportive. Hence, we rate our design as successful. Especially the enhanced spiral plot turned out to be particularly supporting for the detection of anomalies and false positives – crucial tasks in triage analysis. Supporting visual cues resulted in an optimal recognition rate for abnormal high values. With the additional information provided in the enhanced time slider, the participants were able to spot irregularities in the sensor readings directly from the time slider, which is not supported in the original system.

In line with Sweller [188], we believe that extraneous load, that is the amount of cognitive load evoked by system design, depends strongly on the user's previously established knowledge. In order to adjust the system design accordingly, further insights regarding the variation of cognitive load across different tasks are needed. This could also bring further insights regarding tasks that require “too many different steps” (Q2) in the enhanced system. Thus, continuously assessing cognitive load is a research opportunity, e.g., via changes in pupil diameter as described by Duchowski et al. [49]. In addition, gaze patterns could be tracked, to analyze how system features influence and support the user's solution strategies. In particular, the impact of the currently implemented and further visual cues and other guidance options are interesting directions.

Conclusion

Conclusion

Uncertainty is omnipresent when analyzing data, and sometimes difficult to handle and incorporate in visualizations. With this thesis, I contribute to this active field of research in different settings.



I presented new visualization approaches for the topological structure of scalar field ensembles and time-varying scalar fields. Relying on an adaptable, topology-based matching, these approaches provide high flexibility and a novel simultaneous layout for multiple contour trees. Linking to individual ensemble members reveals parameter dependencies.



With a new data structure for vector field ensembles, the fast generation of visitation maps is now possible for ensembles whose size until now prohibited interactive exploration with visitation maps. In addition, the data structure can be used as compression with optimal results in visitation map generation.



Triage analysis in operational technology networks is now supported by a knowledge assisted system combining anomaly detection results with monitoring of device readings. For the first time, a general framework provides guidelines for the incorporation of knowledge assistance in visualization systems.

I presented different applications for all introduced methods; still, there is a wide range of applications in science, engineering and cyber security where my research is applicable and contributes to future development and research.



Both application areas of Fuzzy Contour Trees –scalar field ensembles and time-varying scalar fields– can be found in climate or geology related data. In this case, adapting the metric in the alignment process to application specific metrics, or even change the tracked features to application specific features is possible. In general, our progress on simultaneous layouts of multiple contour trees is beneficial in all scenarios involving several contour trees. While we already considered time-dependent scalar fields, also the comparison of topological

patterns in different locations or the comparison of the characteristics of different samples is possible. Also in situ visualization, allowing an overview of the topological structure of results during their creation is an area of research that becomes available now.



Visitation maps are used in medicine and biology for fibre tracking; encountering data sets that limit the application of visitation maps, Visitation Graphs can provide support and improvements. In addition, the representation of vector field ensembles as their corresponding Visitation Graphs provides a new data structure with many opportunities. Further processing and analysis is possible using graph algorithms. Clustering, comparison and interpolation is possible from a whole new perspective and with different instruments. Also the generalization of Visitation Graphs to three dimensions is a promising research opportunity including research in space partitioning and visualization of visitation maps in three dimensions.



The Knowledge Rocks Framework can be applied to general visualization systems to support the effective reactivation of software resource in the visualization community. Even an application to other systems presented in this thesis is possible, for example to the Fuzzy Contour Trees. In general, knowledge assistance is an active field of research with challenging questions whose future solutions can be implemented and made accessible via the Knowledge Rocks Framework, boosting the prompt and frequent application of current research.

While the Security in Process System is designed for its specific application in operational technology networks, anomaly detection in general device readings is an active area of research whose results can be coupled with the presented visualization and used to develop it further. Additional research opportunities lie in the field of Human Computer Interaction: besides more elaborate evaluation strategies and research on solution strategies using the system, a fundamental link between the visualization of readings and anomaly detection results on one side, and reality (e.g. machine locations, inter-device relations) on the other side is promising. Solving this using augmented or virtual reality in a collaborative setting creates many research questions, challenges and opportunities, also concerning knowledge integration in the system.

In face of their wide range of applications in differing areas, all visualization systems I developed during my research provide unique and sensible solutions for **comparison**, **combination** and **separation** of data and provide a high-level overview of the data and uncertainty structures. I published my work in different journals and presented it in conference and invited talks.

Publications

- [*4]I. Albert. “Context-Creating Visualization of 3D Vector Fields”. Supervisor Anna-Pia Lohfink. MA thesis. 2019 (cit. on p. 87).
- [*6]S. D. D. Anton, A.-P. Lohfink, C. Garth, and H. D. Schotten. “Security in Process: Detecting Attacks in Industrial Process Data”. In: *CECC 2019* (2019). DOI: 10.1145/3360664.3360669 (cit. on pp. 103, 105).
- [*7]S. D. D. Anton, A.-P. Lohfink, and H. D. Schotten. “Discussing the Feasibility of Acoustic Sensors for Side Channel-Aided Industrial Intrusion Detection: An Essay”. In: *CECC 2019* (2019). DOI: 10.1145/3360664.3360667 (cit. on p. 103).
- [*101]A. Keksel, A.-P. Lohfink, M. Eifler, C. Garth, and J. Seewig. “Virtual topography measurement with transfer functions derived by fitted time series models”. In: *Measurement Science and Technology* 31.5 (Oct. 2019), p. 055008. DOI: 10.1088/1361-6501/ab5131 (cit. on p. 6).
- [*115]A.-P. Lohfink, S. D. D. Anton, H. Leitte, and C. Garth. “Knowledge Rocks: Adding Knowledge Assistance to Visualization Systems”. In: *IEEE Transactions on Visualization and Computer Graphics* (2021). DOI: 10.1109/TVCG.2021.3114687 (cit. on pp. 123, 142).
- [*116]A.-P. Lohfink, S. D. D. Anton, H. D. Schotten, H. Leitte, and C. Garth. “Security in Process: Visually Supported Triage Analysis in Industrial Process Data”. In: *IEEE Transactions on Visualization and Computer Graphics* 26.4 (2020). **Best Paper Award VizSec 2019**, pp. 1638–1649. DOI: 10.1109/TVCG.2020.2969007 (cit. on pp. 103, 115, 159).
- [*117]A.-P. Lohfink and C. Garth. “Visitation Graphs: Interactive Ensemble Visualization with Visitation Maps”. In: *2nd International Conference of the DFG International Research Training Group 2057 – Physical Modeling for Virtual Manufacturing (iPMVM 2020)*. Vol. 89. Open Access Series in Informatics (OASICs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 4:1–4:20. DOI: 10.4230/OASICs.iPMVM.2020.4 (cit. on pp. 70, 86, 87).
- [*118]A.-P. Lohfink, F. Gartzky, F. Wetzels, L. Vollmer, and C. Garth. “Time-Varying Fuzzy Contour Trees”. In: *2021 IEEE Visualization Conference (VIS)* (2021), pp. 86–90. DOI: 10.1109/VIS49827.2021.9623286 (cit. on pp. 51, 56).
- [*119]A.-P. Lohfink, V. M. Memmersheimer, F. Gartzky, and C. Garth. “The Enhanced Security in Process System - Evaluating Knowledge Assistance”. In: *2021 IEEE Workshop on TRust and EXPertise in Visual Analytics (TRES)* (2021), pp. 1–7. DOI: 10.1109/TRES53765.2021.00006 (cit. on p. 159).
- [*120]A.-P. Lohfink, F. Wetzels, J. Lukaszcyk, G. H. Weber, and C. Garth. “Fuzzy Contour Trees: Alignment and Joint Layout of Multiple Contour Trees”. In: *Computer Graphics Forum* 39.3 (2020), pp. 343–355. DOI: 10.1111/cgf.13985 (cit. on pp. 14, 40).

Bibliography

- [1] A. Agranovsky, D. Camp, C. Garth, E. W. Bethel, K. I. Joy, and H. Childs. “Improved post hoc flow analysis via Lagrangian representations”. In: *2014 IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV)*. Nov. 2014, pp. 67–75. DOI: 10.1109/LDAV.2014.7013206 (cit. on p. 72).
- [2] J. P. Ahrens, S. Jourdain, P. O’Leary, J. Patchett, D. H. Rogers, and M. Petersen. “An Image-Based Approach to Extreme Scale in Situ Visualization and Analysis”. In: *International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2014, New Orleans, LA, USA, November 16-21, 2014*. Ed. by T. Damkroger and J. J. Dongarra. IEEE Computer Society, 2014, pp. 424–434. DOI: 10.1109/SC.2014.40 (cit. on pp. 50, 75).
- [3] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of time-oriented data*. Springer Science & Business Media, 2011. DOI: 10.1007/978-0-85729-079-3 (cit. on p. 101).
- [5] N. Anh Huynh, W. Keong Ng, A. Ulmer, and J. Kohlhammer. “Uncovering periodic network signals of cyber attacks”. In: *2016 IEEE Symposium on Visualization for Cyber Security (VizSec)*. Oct. 2016, pp. 1–8. DOI: 10.1109/VIZSEC.2016.7739581 (cit. on p. 102).
- [8] G. A. Aranda-Corral, J. Borrego-Díaz, and A. Jiménez-Mavillard. “Social Ontology Documentation for Knowledge Externalization”. In: *Metadata and Semantic Research*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 137–148. DOI: 10.1007/978-3-642-16552-8_14 (cit. on p. 131).
- [9] S. G. Archambault, J. Helouvy, B. Strohl, and G. Williams. “Data visualization as a communication tool”. In: *Library Hi Tech News* 32.2 (2015), pp. 1–9. DOI: 10.1108/LHTN-10-2014-0098 (cit. on p. 111).
- [10] T. Athawale, D. Maljovec, C. R. Johnson, V. Pascucci, and B. Wang. *Uncertainty Visualization of 2D Morse Complex Ensembles Using Statistical Summary Maps*. 2019. arXiv: 1912.06341 [cs.GR] (cit. on p. 49).
- [11] C. L. Bajaj, V. Pascucci, and D. R. Schikore. “The contour spectrum”. In: *Proceedings. Visualization ’97 (Cat. No. 97CB36155)*. 1997, pp. 167–173. DOI: 10.1109/VISUAL.1997.663875 (cit. on p. 53).
- [12] C. Bajaj, A. Gillette, and S. Goswami. “Topology Based Selection and Curation of Level Sets”. In: *Topology-Based Methods in Visualization II*. Mathematics and Visualization. Springer, 2009, pp. 45–58. DOI: 10.1007/978-3-540-88606-8_4 (cit. on p. 16).
- [13] U. Bauer, B. Di Fabio, and C. Landi. “An Edit Distance for Reeb Graphs”. In: *Proceedings of the Eurographics 2016 Workshop on 3D Object Retrieval*. 3DOR ’16. Lisbon, Portugal: Eurographics Association, 2016, pp. 27–34. DOI: 10.2312/3dor.20161084 (cit. on p. 19).
- [14] U. Bauer, X. Ge, and Y. Wang. “Measuring Distance between Reeb Graphs”. In: *Proceedings of the Annual Symposium on Computational Geometry* (July 2013). DOI: 10.1145/2582112.2582169 (cit. on p. 19).

- [15] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. *OWL Web Ontology Language Reference*. <https://www.w3.org/TR/owl-ref/>. Accessed: 2020-11-25 (cit. on p. 131).
- [16] K. Beketayev, D. Yeliussizov, D. Morozov, G. Weber, and B. Hamann. “Measuring the Distance Between Merge Trees”. In: *Topological Methods in Data Analysis and Visualization III*. Springer, Jan. 2014, pp. 151–165. DOI: 10.1007/978-3-319-04099-8_10 (cit. on p. 19).
- [17] J. Benesty, J. Chen, Y. Huang, and I. Cohen. *Noise Reduction in Speech Processing*. Vol. 2. Springer Science & Business Media, 2009, pp. 1–4. DOI: 10.1007/978-3-642-00296-0 (cit. on p. 106).
- [18] J. C. Bennett, V. Krishnamoorthy, S. Liu, R. W. Grout, E. R. Hawkes, J. H. Chen, J. Shepherd, V. Pascucci, and P. Bremer. “Feature-Based Statistical Analysis of Combustion Simulation Data”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 1822–1831. DOI: 10.1109/TVCG.2011.199 (cit. on p. 54).
- [19] W. Berger, H. Piringer, P. Filzmoser, and E. Gröller. “Uncertainty-Aware Exploration of Continuous Parameter Spaces Using Multivariate Prediction”. In: *Computer Graphics Forum* 30.3 (2011), pp. 911–920. DOI: 10.1111/j.1467-8659.2011.01940.x (cit. on p. 8).
- [20] P. Bille. “A survey on tree edit distance and related problems”. In: *Theoretical Computer Science* 337.1-3 (2005), pp. 217–239. DOI: 10.1016/j.tcs.2004.12.030 (cit. on pp. 19–23).
- [21] R. A. Boller, S. A. Braun, J. Miles, and D. H. Laidlaw. “Application of uncertainty visualization methods to meteorological trajectories”. In: *Earth Science Informatics* 3.1 (June 2010), pp. 119–126. DOI: 10.1007/s12145-010-0052-5 (cit. on pp. 8, 72).
- [22] 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: Uncertainty, Multifield, Biomedical, and Scalable Visualization*. London: Springer London, 2014, pp. 3–27. DOI: 10.1007/978-1-4471-6497-5_1 (cit. on p. 7).
- [23] A. Boschetti, L. Salgarelli, C. Muelder, and K.-L. Ma. “TVi: A Visual Querying System for Network Monitoring and Anomaly Detection”. In: *Proceedings of the 8th International Symposium on Visualization for Cyber Security*. VizSec ’11. Pittsburgh, Pennsylvania, USA: ACM, 2011, 1:1–1:10. DOI: 10.1145/2016904.2016905. URL: <http://doi.acm.org/10.1145/2016904.2016905> (cit. on p. 103).
- [24] M. Bostock, V. Ogievetsky, and J. Heer. “D³ Data-Driven Documents”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (Dec. 2011), pp. 2301–2309. DOI: 10.1109/TVCG.2011.185 (cit. on pp. 33, 153).
- [25] P.-T. Bremer, E. M. Bringa, M. A. Duchaineau, A. G. Gyulassy, D. Laney, A. Mascarenhas, and V. Pascucci. “Topological feature extraction and tracking”. In: *Journal of Physics: Conference Series* 78 (July 2007), p. 012007. DOI: 10.1088/1742-6596/78/1/012007 (cit. on p. 54).

- [26]P.-T. Bremer, G. Weber, V. Pascucci, M. Day, and J. Bell. “Analyzing and Tracking Burning Structures in Lean Premixed Hydrogen Flames”. In: *IEEE Transactions on Visualization and Computer Graphics* 16 (May 2010), pp. 248–60. DOI: 10.1109/TVCG.2009.69 (cit. on pp. 16, 18).
- [27]K. Brodlie, R. Allendes Osorio, and A. Lopes. “A Review of Uncertainty in Data Visualization”. In: *Expanding the Frontiers of Visual Analytics and Visualization*. London: Springer London, 2012, pp. 81–109. DOI: 10.1007/978-1-4471-2804-5_6 (cit. on pp. 1, 6).
- [28]K. Bürger, R. Fraedrich, D. Merhof, and R. Westermann. “Instant visitation maps for interactive visualization of uncertain particle trajectories”. In: *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics. 2012, 82940P–82940P (cit. on pp. 72, 73).
- [29]T. Buzan and B. Buzan. *The mind map book*. Pearson Education, 2006 (cit. on p. 131).
- [30]S. Carpendale, M. Chen, D. Evanko, N. Gehlenborg, C. Görg, L. Hunter, F. Rowland, M. Storey, and H. Strobel. “Ontologies in Biological Data Visualization”. In: *IEEE Computer Graphics and Applications* 34.2 (2014), pp. 8–15. DOI: 10.1109/MCG.2014.33 (cit. on pp. 127, 131).
- [31]H. Carr, J. Snoeyink, and M. van de Panne. “Simplifying flexible isosurfaces using local geometric measures”. In: *IEEE Visualization 2004*. Oct. 2004, pp. 497–504. DOI: 10.1109/VISUAL.2004.96 (cit. on pp. 16, 26).
- [32]H. Carr and D. Duke. “Joint Contour Nets”. In: *IEEE Transactions on Visualization and Computer Graphics* 20.8 (Dec. 2013), pp. 1100–1113. DOI: 10.1109/TVCG.2013.269 (cit. on p. 16).
- [33]H. Carr, J. Snoeyink, and U. Axen. “Computing contour trees in all dimensions”. In: *Computational Geometry* 24.2 (2003). Special Issue on the Fourth CGC Workshop on Computational Geometry, pp. 75–94. DOI: 10.1016/S0925-7721(02)00093-7 (cit. on p. 16).
- [34]M. Carrière and S. Oudot. “Local Equivalence and Intrinsic Metrics between Reeb Graphs”. In: *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*. Ed. by B. Aronov and M. J. Katz. Vol. 77. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 25:1–25:15. DOI: 10.4230/LIPIcs.SoCG.2017.25 (cit. on p. 19).
- [35]M. Caselli, E. Zambon, and F. Kargl. “Sequence-aware intrusion detection in industrial control systems”. In: ACM. 2015, pp. 13–24. DOI: 10.1145/2732198.2732200 (cit. on p. 110).
- [36]D. Ceneda, T. Gschwandtner, T. May, S. Miksch, H. Schulz, M. Streit, and C. Tominski. “Characterizing Guidance in Visual Analytics”. In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 111–120. DOI: 10.1109/TVCG.2016.2598468 (cit. on pp. 125, 133).

- [37]D. Ceneda, N. Andrienko, G. Andrienko, T. Gschwandtner, S. Miksch, N. Piccolotto, T. Schreck, M. Streit, J. Suschnigg, and C. Tominski. “Guide Me in Analysis: A Framework for Guidance Designers”. In: *Computer Graphics Forum* 39.6 (2020), pp. 269–288. DOI: <https://doi.org/10.1111/cgf.14017> (cit. on p. 136).
- [38]D. Ceneda, T. Gschwandtner, S. Miksch, and C. Tominski. *Guided Visual Exploration of Cyclical Patterns in Time-series*. https://publik.tuwien.ac.at/files/publik_272789.pdf. poster presentation: Visualization in Data Science (VDS at IEEE VIS 2018). Oct. 2018 (cit. on p. 155).
- [39]W. L. Chapman and J. E. Walsh. *Arctic and Southern Ocean Sea Ice Concentrations, Version 1, Arctic*. Boulder, Colorado USA. NSIDC: National Snow and Ice Data Center. 1991, updated 1996, Accessed: 2021-02-20. DOI: 10.7265/N5057CVT (cit. on p. 63).
- [40]C. Chen. “Top 10 Unsolved Information Visualization Problems”. In: *IEEE Computer Graphics and Applications* 25.4 (2005), pp. 12–16. DOI: 10.1109/MCG.2005.91 (cit. on p. 123).
- [41]M. Chen, D. Ebert, H. Hagen, R. S. Laramée, R. van Liere, K. -L. Ma, W. Ribarsky, G. Scheuermann, and D. Silver. “Data, Information, and Knowledge in Visualization”. In: *IEEE Computer Graphics and Applications* 29.1 (2009), pp. 12–19. DOI: 10.1109/MCG.2009.6 (cit. on pp. 123, 149).
- [42]M. Chen and H. Hagen. “Guest Editors’ Introduction: Knowledge-Assisted Visualization”. In: *IEEE Computer Graphics and Applications* 30.1 (Jan. 2010), pp. 15–16. DOI: 10.1109/MCG.2010.8 (cit. on p. 122).
- [43]Y.-J. Chiang and X. Lu. “Progressive Simplification of Tetrahedral Meshes Preserving All Isosurface Topologies”. In: *Comput. Graph. Forum* 22.3 (2003), pp. 493–504. DOI: 10.1111/1467-8659.00697 (cit. on p. 16).
- [44]H. Childs. “Data exploration at the exascale”. In: *Supercomputing frontiers and innovations* 2.3 (2015), pp. 5–13 (cit. on p. 75).
- [45]J. Cox, D. House, and M. Lindell. “VISUALIZING UNCERTAINTY IN PREDICTED HURRICANE TRACKS”. In: *International Journal for Uncertainty Quantification* 3.2 (2013), pp. 143–156 (cit. on pp. 8, 72).
- [46]S. Deitrick and R. Edsall. “The Influence of Uncertainty Visualization on Decision Making: An Empirical Evaluation”. In: *Progress in Spatial Data Handling: 12th International Symposium on Spatial Data Handling*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 719–738. DOI: 10.1007/3-540-35589-8_45 (cit. on p. 7).
- [47]I. Demir, C. Dick, and R. Westermann. “Multi-Charts for Comparative 3D Ensemble Visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 2694–2703. DOI: 10.1109/TVCG.2014.2346448 (cit. on p. 133).
- [48]H. Doraiswamy and V. Natarajan. “Efficient algorithms for computing Reeb graphs”. In: *Computational Geometry* 42.6-7 (2009), pp. 606–616. DOI: 10.1016/j.comgeo.2008.12.003 (cit. on p. 16).

- [49]A. T. Duchowski, K. Krejtz, I. Krejtz, C. Biele, A. Niedzielska, P. Kiefer, M. Raubal, and I. Giannopoulos. “The Index of Pupillary Activity: Measuring Cognitive Load Vis-à-Vis Task Difficulty with Pupil Oscillation”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI ’18. Montreal QC, Canada: Association for Computing Machinery, 2018, pp. 1–13. DOI: 10.1145/3173574.3173856. URL: <https://doi.org/10.1145/3173574.3173856> (cit. on p. 171).
- [50]M. Dudáš, S. Lohmann, V. Svátek, and D. Pavlov. “Ontology Visualization Methods and Tools: a Survey of the State of the Art”. In: *The Knowledge Engineering Review* 33 (July 2018). DOI: 10.1017/S0269888918000073 (cit. on p. 134).
- [51]S. Duque Antón, D. Fraunholz, C. Lipps, F. Pohl, M. Zimmermann, and H. D. Schotten. “Two decades of SCADA exploitation: A brief history”. In: *2017 IEEE Conference on Application, Information and Network Security (AINS)*. IEEE, 2017, pp. 98–104 (cit. on p. 102).
- [52]S. Duque Antón, A. Hafner, and H. D. Schotten. “Devil in the Detail: Attack Scenarios in Industrial Applications”. In: *2019 IEEE Security and Privacy Workshops*. IEEE, 2019 (cit. on p. 102).
- [53]S. Dutta, C.-M. Chen, G. Heinlein, H.-W. Shen, and J.-P. Chen. “In situ distribution guided analysis and visualization of transonic jet engine simulations”. In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 811–820 (cit. on p. 75).
- [54]H. Edelsbrunner, J. Harer, A. Mascarenhas, V. Pascucci, and J. Snoeyink. “Time-varying Reeb graphs for continuous space–time data”. In: *Computational Geometry* 41.3 (2008), pp. 149–166. DOI: 10.1016/j.comgeo.2007.11.001 (cit. on p. 53).
- [55]N. Fabian, K. Moreland, D. Thompson, A. C. Bauer, P. Marion, B. Gevecik, M. Rasquin, and K. E. Jansen. “The paraview coprocessing library: A scalable, general purpose in situ visualization library”. In: *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*. IEEE, 2011, pp. 89–96 (cit. on p. 75).
- [56]G. Favelier, N. Faraj, B. Summa, and J. Tierny. “Persistence Atlas for Critical Point Variability in Ensembles”. In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (Jan. 2019), pp. 1152–1162. DOI: 10.1109/TVCG.2018.2864432 (cit. on p. 49).
- [57]P. Federico, M. Wagner, A. Rind, A. Amor-Amorós, S. Miksch, and W. Aigner. “The Role of Explicit Knowledge: A Conceptual Model of Knowledge-Assisted Visual Analytics”. In: *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*. 2017, pp. 92–103. DOI: 10.1109/VAST.2017.8585498 (cit. on pp. 123, 124, 126, 130).
- [58]C. Feng, T. Li, and D. Chana. “Multi-level Anomaly Detection in Industrial Control Systems via Package Signatures and LSTM Networks”. In: *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. June 2017, pp. 261–272. DOI: 10.1109/DSN.2017.34 (cit. on p. 103).

- [59]D. Feng, L. Kwock, Y. Lee, and R. Taylor. “Matching Visual Saliency to Confidence in Plots of Uncertain Data”. In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (Nov. 2010), pp. 980–989. DOI: 10.1109/TVCG.2010.176 (cit. on pp. 8, 72).
- [60]D. Feng, L. Kwock, Y. Lee, and R. M. Taylor. “Linked exploratory visualizations for uncertain MR spectroscopy data”. In: *Proc.SPIE* 7530 (2010), pp. 7530 - 7530–12. DOI: 10.1117/12.839818 (cit. on pp. 8, 72).
- [61]F. Ferstl, K. Bürger, and R. Westermann. “Streamline variability plots for characterizing the uncertainty in vector field ensembles”. In: *IEEE Transactions on Visualization and Computer Graphics* 22.1 (2016), pp. 767–776 (cit. on pp. 72, 73).
- [62]N. Fout and K. L. Ma. “Fuzzy Volume Rendering”. In: *IEEE Transactions on Visualization and Computer Graphics* 18.12 (Dec. 2012), pp. 2335–2344. DOI: 10.1109/TVCG.2012.227 (cit. on p. 8).
- [63]I. Fujishiro, Y. Takeshima, Y. Ichikawa, and K. Nakamura. “GADGET: goal-oriented application design guidance for modular visualization environments”. In: *Proceedings. Visualization '97 (Cat. No. 97CB36155)*. 1997, pp. 245–252. DOI: 10.1109/VISUAL.1997.663889 (cit. on p. 122).
- [64]M. Gaillard. “CERN Data Centre passes the 200-petabyte milestone”. In: *CERN News – Computing* (July 2017). accessed Apr 28 2021. URL: <https://home.cern/news/news/computing/cern-data-centre-passes-200-petabyte-milestone> (cit. on p. 12).
- [65]E. R. Gansner, E. Koutsofios, S. C. North, and K.-P. Vo. “A Technique for Drawing Directed Graphs”. In: *IEEE Trans. Software Eng.* 19.3 (1993), pp. 214–230. DOI: 10.1109/32.221135 (cit. on p. 17).
- [66]N. Gershon. “Visualization of an Imperfect World”. In: *IEEE Computer Graphics and Applications* 18.4 (1998), pp. 43–45. DOI: 10.1109/38.689662 (cit. on p. 7).
- [67]O. Gilson, N. Silva, P. Grant, and M. Chen. “From Web Data to Visualization via Ontology Mapping”. In: *Computer Graphics Forum* 27.3 (2008), pp. 959–966. DOI: 10.1111/j.1467-8659.2008.01230.x (cit. on p. 127).
- [68]M. Gleicher. “Considerations for Visualizing Comparison”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2017), pp. 413–423. DOI: 10.1109/TVCG.2017.2744199 (cit. on p. 133).
- [69]M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts. “Visual Comparison for Information Visualization”. In: *Information Visualization* 10.4 (2011), pp. 289–309. DOI: 10.1177/1473871611416549 (cit. on p. 133).
- [70]J. Goh, S. Adepu, K. N. Junejo, and A. Mathur. “A Dataset to Support Research in the Design of Secure Water Treatment Systems”. In: *Critical Information Infrastructures Security*. Ed. by G. Havarneanu, R. Setola, H. Nassopoulos, and S. Wolthusen. Cham: Springer International Publishing, 2017, pp. 88–99. DOI: 10.1007/978-3-319-71368-7_8 (cit. on p. 114).

- [71]J. Goh, S. Adepu, M. Tan, and Z. S. Lee. “Anomaly Detection in Cyber Physical Systems Using Recurrent Neural Networks”. In: *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*. Jan. 2017, pp. 140–145. DOI: 10.1109/HASE.2017.36 (cit. on p. 103).
- [72]R. Gove and L. Deason. “Visualizing Automatically Detected Periodic Network Activity”. In: *2018 IEEE Symposium on Visualization for Cyber Security (VizSec)*. IEEE. Oct. 2018, pp. 1–8. DOI: 10.1109/VIZSEC.2018.8709177 (cit. on p. 102).
- [73]Y. Gu and C. Wang. “Transgraph: Hierarchical exploration of transition relationships in time-varying volumetric data”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 2015–2024 (cit. on p. 75).
- [74]D. Günther, J. Salmon, and J. Tierny. “Mandatory Critical Points of 2D Uncertain Scalar Fields”. In: *Computer Graphics Forum (Proc. EuroVis)* 33 (June 2014), pp. 31–40. DOI: 10.1111/cgf.12359 (cit. on p. 19).
- [75]R. Haimes. “pV3-A distributed system for large-scale unsteady CFD visualization”. In: *32nd Aerospace Sciences Meeting and Exhibit*. 1994, p. 321 (cit. on p. 75).
- [76]M. Hajij and P. A. Rosen. “An Efficient Data Retrieval Parallel Reeb Graph Algorithm”. In: *Algorithms* 13.10 (2020), p. 258. DOI: 10.3390/a13100258 (cit. on p. 16).
- [77]D. J. Hand. “Intelligent data analysis: Issues and opportunities”. In: *Advances in Intelligent Data Analysis Reasoning about Data*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 1–14 (cit. on p. 149).
- [78]S. G. Hart. “Nasa-Task Load Index (NASA-TLX); 20 Years Later”. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50.9 (2006), pp. 904–908. DOI: 10.1177/154193120605000909 (cit. on p. 169).
- [79]C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. De Florian, G. Scheuermann, and C. Garth. “A Survey of Topology-based Methods in Visualization”. In: *Computer Graphics Forum* 35 (June 2016), pp. 643–667. DOI: 10.1111/cgf.12933 (cit. on p. 53).
- [80]C. Heine, D. Schneider, H. Carr, and G. Scheuermann. “Drawing Contour Trees in the Plane”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.11 (Nov. 2011), pp. 1599–1611. DOI: 10.1109/TVCG.2010.270 (cit. on pp. 17, 19, 27, 29, 34).
- [81]M. Hellkamp et al. *Bottle: Python Web Framework*. <https://bottlepy.org/docs/dev/>. software, version 0.12. 2016 (cit. on p. 153).
- [82]M. Hlawatsch, P. Leube, W. Nowak, and D. Weiskopf. “Flow Radar Glyphs - Static Visualization of Unsteady Flow with Uncertainty”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 1949–1958 (cit. on pp. 2, 8, 72).
- [83]C. M. Hoffmann and M. J. O’Donnell. “Pattern Matching in Trees”. In: *J. ACM* 29.1 (1982), pp. 68–95. DOI: 10.1145/322290.322295 (cit. on p. 19).

- [84]L. Hofmann and F. Sadlo. “Extraction of Distinguished Hyperbolic Trajectories for 2D Time-Dependent Vector Field Topology”. In: *Comput. Graph. Forum* 39.3 (2020), pp. 303–315. DOI: 10.1111/cgf.13982. URL: <https://doi.org/10.1111/cgf.13982> (cit. on p. 53).
- [85]P. Hristov and H. Carr. “W-Structures in Contour Trees”. In: *Topological Methods in Visualization: Theory, Software and Applications*. 2020 (cit. on p. 20).
- [86]W. Hu, T. Chen, and S. L. Shah. “Detection of Frequent Alarm Patterns in Industrial Alarm Floods Using Itemset Mining Methods”. In: *IEEE Transactions on Industrial Electronics* 65.9 (Sept. 2018), pp. 7290–7300. DOI: 10.1109/TIE.2018.2795573 (cit. on p. 103).
- [87]M. Hummel, H. Obermaier, C. Garth, and K. I. Joy. “Comparative Visual Analysis of Lagrangian Transport in CFD Ensembles”. In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (Dec. 2013), pp. 2743–2752. DOI: 10.1109/TVCG.2013.141 (cit. on pp. 8, 72).
- [88]N. A. Huynh, W. K. Ng, and H. G. Do. “On periodic behavior of malware: experiments, opportunities and challenges”. In: *2016 11th International Conference on Malicious and Unwanted Software (MALWARE)*. IEEE, 2016, pp. 1–8. DOI: 10.1109/MALWARE.2016.7888733 (cit. on p. 102).
- [89]V. M. Ijure, S. A. Laughter, and R. D. Williams. “Security issues in SCADA networks”. In: *computers & security* 25.7 (2006), pp. 498–506. DOI: 10.1016/j.cose.2006.03.001 (cit. on p. 102).
- [90]T. Ishikawa, K. Kobayashi, M. Okabe, and T. Yamaguchi. “Support for externalization of intelligence skill using ontology and rule-based system”. English. In: *Proceedings of the 9th Joint Conference on Knowledge-Based Software Engineering, JCKBSE 2010*. Proceedings of the 9th Joint Conference on Knowledge-Based Software Engineering, JCKBSE 2010. 9th Joint Conference on Knowledge-Based Software Engineering, JCKBSE 2010 ; Conference date: 25-08-2010 Through 27-08-2010. Dec. 2010, pp. 145–159 (cit. on p. 131).
- [91]iTrust Centre for Research in Cyber Security. *Secure Water Treatment (SWaT) Testbed*. Tech. rep. 4.2. Singapore University of Technology and Design, Oct. 2018 (cit. on p. 104).
- [92]H. Jänicke and G. Scheuermann. “Visual analysis of flow features using information theory”. In: *IEEE Computer Graphics and Applications* 30.1 (2010), pp. 40–49 (cit. on p. 75).
- [93]M. Jarema, J. Kehrer, and R. Westermann. “Comparative visual analysis of transport variability in flow ensembles”. In: *Journal of WSCG*. 24.1 (2016), pp. 25–34. URL: <http://hdl.handle.net/11025/21642> (cit. on pp. 8, 72).
- [94]*JavaScript Semantic Web Toolkit Project Page*. <https://code.google.com/archive/p/owlreasoner/>. Accessed: 2020-03-29 (cit. on p. 138).

- [95]S. Jiang, S. Fang, S. Bloomquist, J. Keiper, M. J. Palakal, Y. Xia, and S. J. Grannis. “Healthcare Data Visualization: Geospatial and Temporal Integration”. In: *VISIGRAPP (2: IVAPP)*. Jan. 2016, pp. 212–219. DOI: 10.5220/0005714002120219 (cit. on p. 154).
- [96]T. Jiang, L. Wang, and K. Zhang. “Alignment of Trees - An Alternative to Tree Edit”. In: *Combinatorial Pattern Matching, 5th Annual Symposium, CPM 94, Asilomar, California, USA, June 5-8, 1994, Proceedings*. 1994, pp. 75–86. DOI: 10.1007/3-540-58094-8_7 (cit. on p. 19).
- [97]D. K. Jones. “Tractography Gone Wild: Probabilistic Fibre Tracking Using the Wild Bootstrap With Diffusion Tensor MRI”. In: *IEEE Transactions on Medical Imaging* 27.9 (Sept. 2008), pp. 1268–1274. DOI: 10.1109/TMI.2008.922191 (cit. on p. 72).
- [98]D. K. Jones and C. Pierpaoli. “Confidence mapping in diffusion tensor magnetic resonance imaging tractography using a bootstrap approach”. In: *Magnetic Resonance in Medicine* 53.5 (2005), pp. 1143–1149. DOI: 10.1002/mrm.20466 (cit. on p. 72).
- [99]C. P. Kappe, M. Böttinger, and H. Leitte. “Topology-based Feature Detection in Climate Data”. In: *Workshop on Visualisation in Environmental Sciences (EnvirVis)*. The Eurographics Association, 2019. DOI: 10.2312/envirvis.20191099 (cit. on p. 53).
- [100]E. Karapistoli, P. Sarigiannidis, and A. A. Economides. “SRNET: A Real-time, Cross-based Anomaly Detection and Visualization System for Wireless Sensor Networks”. In: *Proceedings of the Tenth Workshop on Visualization for Cyber Security. VizSec '13*. Atlanta, Georgia, USA: ACM, 2013, pp. 49–56. DOI: 10.1145/2517957.2517964. URL: <http://doi.acm.org/10.1145/2517957.2517964> (cit. on p. 103).
- [102]L. Kettner, J. Rossignac, and J. Snoeyink. “The Safari interface for visualizing time-dependent volume data using iso-surfaces and contour spectra”. In: *Computational Geometry* 25.1 (2003). European Workshop on Computational Geometry - CG01, pp. 97–116. DOI: 10.1016/S0925-7721(02)00132-3 (cit. on pp. 16, 53).
- [103]R. Kincaid and H. Lam. “Line Graph Explorer: Scalable Display of Line Graphs Using Focus+Context”. In: *Proceedings of the Working Conference on Advanced Visual Interfaces. AVI '06*. Venezia, Italy: Association for Computing Machinery, 2006, pp. 404–411. DOI: 10.1145/1133265.1133348 (cit. on p. 153).
- [104]A. D. Kiureghian and O. Ditlevsen. “Aleatory or epistemic? Does it matter?” In: *Structural Safety* 31.2 (2009). Risk Acceptance and Risk Communication, pp. 105–112. DOI: 10.1016/j.strusafe.2008.06.020 (cit. on p. 6).
- [105]P. N. Klein, S. Tirthapura, D. Sharvit, and B. B. Kimia. “A Tree-edit-distance Algorithm for Comparing Simple, Closed Shapes”. In: *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA*. 2000, pp. 696–704 (cit. on p. 19).
- [106]M. Kraus. “Visualization of Uncertain Contour Trees.” In: *IMAGAPP/IVAPP*. Jan. 2010, pp. 132–139 (cit. on p. 18).

- [107] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova. “Compressing the Incompressible with ISABELA: In-situ Reduction of Spatio-temporal Data”. In: *Euro-Par 2011 Parallel Processing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 366–379 (cit. on p. 75).
- [108] H. S. Lallie, L. A. Shepherd, J. R. C. Nurse, A. Erola, G. Epiphaniou, C. Maple, and X. J. A. Bellekens. “Cyber security in the age of COVID-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic”. In: *Comput. Secur.* 105 (2021), p. 102248. DOI: 10.1016/j.cose.2021.102248 (cit. on p. 102).
- [109] J.-B. Lamy. *Owlready2 Documentation*. <https://owlready2.readthedocs.io/en/latest/>. Accessed: 2020-11-27 (cit. on p. 138).
- [110] B. Lee, G. Robertson, M. Czerwinski, and C. Parr. “CandidTree: Visualizing Structural Uncertainty in Similar Hierarchies”. In: *Information Visualization* 6 (Oct. 2007). DOI: 10.1057/palgrave.ivs.9500157 (cit. on p. 18).
- [111] T. Lee, X. Tong, H. Shen, P. C. Wong, S. Hagos, and L. R. Leung. “Feature Tracking and Visualization of the Madden-Julian Oscillation in Climate Simulation”. In: *IEEE Computer Graphics and Applications* 33.4 (2013), pp. 29–37. DOI: 10.1109/MCG.2013.36 (cit. on p. 54).
- [112] H. Lehmann and B. Jung. “In-situ multi-resolution and temporal data compression for visual exploration of large-scale scientific simulations”. In: *Large Data Analysis and Visualization (LDAV), 2014 IEEE 4th Symposium on*. IEEE, 2014, pp. 51–58 (cit. on p. 75).
- [113] S. Li, N. Marsaglia, C. Garth, J. Woodring, J. Clyne, and H. Childs. “Data Reduction Techniques for Simulation, Visualization and Data Analysis”. In: *Computer Graphics Forum* 37.6 (2018), pp. 422–447. DOI: 10.1111/cgf.13336 (cit. on p. 75).
- [114] J. F. Lofstead, S. Klasky, K. Schwan, N. Podhorszki, and C. Jin. “Flexible IO and integration for scientific codes through the adaptable IO system (ADIOS)”. In: *Proceedings of the 6th international workshop on Challenges of large applications in distributed environments*. ACM, 2008, pp. 15–24 (cit. on p. 75).
- [121] A.-P. O. Lohfink, N. S. Ströhmer-Lohfink, and M. S. Schubert. “Under View Control”. In: *Evoke as SubGround4* (2013). URL: <http://www.pouet.net/prod.php?which=61750> (cit. on p. 43).
- [122] A. L. Love, A. Pang, and D. L. Kao. “Visualizing spatial multivalued data”. In: *IEEE Computer Graphics and Applications* 25.3 (May 2005), pp. 69–79. DOI: 10.1109/MCG.2005.71 (cit. on p. 72).
- [123] J. Lukasczyk, G. Aldrich, M. Steptoe, G. Favelier, C. Gueunet, J. Tierny, R. Maciejewski, B. Hamann, and H. Leitte. “Viscous Fingering: A Topological Visual Analytic Approach”. In: *Applied Mechanics and Materials*. Vol. 869. Trans Tech Publ, 2017, pp. 9–19. DOI: 10.4028/www.scientific.net/AMM.869.9 (cit. on p. 47, 54).

- [124]J. Lukasczyk, R. Maciejewski, C. Garth, and H. Hagen. “Understanding Hotspots: A Topological Visual Analytics Approach”. In: *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. Association for Computing Machinery, 2015. DOI: 10.1145/2820783.2820817 (cit. on pp. 53, 54).
- [125]J. Lukasczyk, G. Weber, R. Maciejewski, C. Garth, and H. Leitte. “Dynamic Nested Tracking Graphs”. In: *Computer Graphics Forum* 36.3 (2017), pp. 12–22. DOI: 10.1111/cgf.13164 (cit. on pp. 18, 54).
- [126]J. Ma, C. Wang, and C.-K. Shene. “FlowGraph: A compound hierarchical graph for flow field exploration”. In: *2013 IEEE Pacific Visualization Symposium (PacificVis)*. IEEE. 2013, pp. 233–240 (cit. on p. 74).
- [127]A. M. MacEachren. “Accuracy of Thematic Maps / Implications of Choropleth Symbolization”. In: *Cartographica: The International Journal for Geographic Information and Geovisualization* 22.1 (1985), pp. 38–58. DOI: 10.3138/Y481-6J14-8802-G755 (cit. on p. 7).
- [128]L. A. Maglaras and J. Jiang. “Intrusion Detection in SCADA Systems Using Machine Learning Techniques”. In: *2014 Science and Information Conference*. IEEE. 2014, pp. 626–631 (cit. on p. 103).
- [129]A. P. Mathur and N. O. Tippenhauer. “SWaT: a water treatment testbed for research and training on ICS security”. In: *2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*. Apr. 2016, pp. 31–36. DOI: 10.1109/CySWater.2016.7469060 (cit. on p. 104).
- [130]*Matrix Profiles Interactive Example*. <https://ui.matrixprofile.org/>. Accessed: 2019-07-28 (cit. on p. 105).
- [131]T. McLoughlin, R. S. Laramée, R. Peikert, F. H. Post, and M. Chen. “Over Two Decades of Integration-Based, Geometric Flow Visualization”. In: *Comput. Graph. Forum* 29.6 (2010), pp. 1807–1829. DOI: 10.1111/j.1467-8659.2010.01650.x (cit. on p. 71).
- [132]S. Miksch, H. Leitte, and M. Chen. “Knowledge-Assisted Visualization and Guidance”. In: *Foundations of Data Visualization*. Cham: Springer International Publishing, 2020, pp. 61–85. DOI: 10.1007/978-3-030-34444-3_4 (cit. on pp. 123, 127).
- [133]M. Mirzargar, R. T. Whitaker, and R. M. Kirby. “Curve boxplot: Generalization of boxplot for ensembles of curves”. In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 2654–2663 (cit. on pp. 8, 72).
- [134]Modbus Organization. *MODBUS Application Protocol Specification V1.1b3*. http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf. Accessed: 2019-08-23. Apr. 2012 (cit. on p. 102).
- [135]Modbus Organization. *MODBUS Messaging on TCP/IP Implementation Guide V1.0b*. http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf. Accessed: 2019-08-23. Oct. 2006 (cit. on p. 102).

- [136]M. Morato-Moreno. “Origins of the Two-Dimensional Relief Representation on some Spanish American Maps in the Sixteenth Century”. In: *Boletin de la Asociacion de Geografos Espanoles* 73 (2017), pp. 493–499. DOI: 10.21138/bage.2414 (cit. on p. 15).
- [137]D. Moritz, C. Wang, G. L. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer. “Formalizing Visualization Design Knowledge as Constraints: Actionable and Extensible Models in Draco”. In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2019), pp. 438–448. DOI: 10.1109/TVCG.2018.2865240 (cit. on pp. 126, 137).
- [138]D. Morozov, K. Beketayev, and G. Weber. “Interleaving Distance between Merge Trees”. In: *Discrete and Computational Geometry* 49 (Jan. 2013), pp. 22–45 (cit. on p. 19).
- [139]A. Mueen, S. Nath, and J. Liu. “Fast Approximate Correlation for Massive Time-series Data”. In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’10. Indianapolis, Indiana, USA: ACM, 2010, pp. 171–182. DOI: 10.1145/1807167.1807188 (cit. on p. 106).
- [140]R. Munroe. *Error Bars*. Under Creative Commons Attribution-NonCommercial 2.5 License. URL: <https://xkcd.com/2110/> (cit. on p. 2).
- [141]L. Muthuppalaniappan Menaka and K. Stevenson. “Healthcare cyber-attacks and the COVID-19 pandemic: an urgent threat to global health”. In: *International Journal for Quality in Health Care* 33.1 (Sept. 2020). DOI: 10.1093/intqhc/mzaa117 (cit. on p. 102).
- [142]National Aeronautics and Space Administration. *NASA TLX Paper and Pencil Version Instruction Manual*. <https://humansystems.arc.nasa.gov/groups/tlx/tlxpaperpencil.php>. questionnaire, accessed 06/25/2021 (cit. on p. 169).
- [143]K. Nie, P. Baltzer, B. Preim, and G. Mistelbauer. “Knowledge-Assisted Comparative Assessment of Breast Cancer using Dynamic Contrast-Enhanced Magnetic Resonance Imaging”. In: *Computer Graphics Forum* 39.3 (2020), pp. 13–23. DOI: 10.1111/cgf.13959 (cit. on p. 126).
- [144]I. Nonaka and H. Takeuchi. “The knowledge-creating company”. In: *Harvard business review* 85.7/8 (2007), p. 162 (cit. on p. 124).
- [145]B. Nouanesengsy, T.-Y. Lee, and H.-W. Shen. “Load-balanced parallel streamline generation on large scale vector fields”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 1785–1794 (cit. on p. 73).
- [146]J. D. Novak, D. B. Gowin, and G. D. Bob. *Learning how to learn*. cambridge University press, 1984 (cit. on p. 131).
- [147]N. F. Noy, M. Crubézy, R. W. Fergerson, H. Knublauch, S. W. Tu, J. Vendetti, and M. A. Musen. “Protégé-2000: An Open-Source Ontology-Development and Knowledge-Acquisition Environment”. In: *AMIA... Annual Symposium proceedings. AMIA Symposium*. Feb. 2003, p. 953 (cit. on p. 138).

- [148]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 IV*. Springer International Publishing, 2017, pp. 87–101 (cit. on p. 53).
- [149]M. Otto, T. Germer, and H. Theisel. “Uncertain topology of 3D vector fields”. In: *2011 IEEE Pacific Visualization Symposium*. Mar. 2011, pp. 67–74. DOI: 10.1109/PACIFICVIS.2011.5742374 (cit. on pp. 8, 72).
- [150]M. Otto, T. Germer, H.-C. Hege, and H. Theisel. “Uncertain 2D Vector Field Topology”. In: *Computer Graphics Forum* 29.2 (2010), pp. 347–356. DOI: 10.1111/j.1467-8659.2009.01604.x (cit. on pp. 8, 72).
- [151]L. M. Padilla, I. T. Ruginski, and S. H. Creem-Regehr. “Effects of ensemble and summary displays on interpretations of geospatial uncertainty data”. In: *Cognitive research: principles and implications* 2.1 (2017), pp. 1–16. DOI: 10.1186/s41235-017-0076-1 (cit. on p. 7).
- [152]V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. “The TOPORRERY: computation and presentation of multi-resolution topology”. In: *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*. Ed. by T. Möller, B. Hamann, and R. D. Russell. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 19–40. DOI: 10.1007/b106657_2 (cit. on p. 13).
- [153]T. Pfaffelmoser, M. Reitinger, and R. Westermann. “Visualizing the Positional and Geometrical Variability of Isosurfaces in Uncertain Scalar Fields”. In: *Computer Graphics Forum* 30.3 (2011), pp. 951–960. DOI: 10.1111/j.1467-8659.2011.01944.x (cit. on p. 8).
- [154]W. A. Pike, J. Stasko, R. Chang, and T. O’connell. “The Science of Interaction”. In: *Information Visualization* 8.4 (Dec. 2009), pp. 263–274. DOI: 10.1057/ivs.2009.22 (cit. on p. 122).
- [155]K. Pothkow and H. C. Hege. “Positional Uncertainty of Isocontours: Condition Analysis and Probabilistic Measures”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.10 (Oct. 2011), pp. 1393–1406. DOI: 10.1109/TVCG.2010.247 (cit. on p. 8).
- [156]K. Potter, P. Rosen, and C. R. Johnson. “From Quantification to Visualization: A Taxonomy of Uncertainty Visualization Approaches”. In: *Uncertainty Quantification in Scientific Computing - 10th IFIP WG 2.5 Working Conference, WoCoUQ 2011, Boulder, CO, USA, August 1-4, 2011, Revised Selected Papers*. Vol. 377. IFIP Advances in Information and Communication Technology. Springer, 2011, pp. 226–249. DOI: 10.1007/978-3-642-32677-6_15 (cit. on p. 7).
- [157]J.-S. Prassni, T. Ropinski, and K. Hinrichs. “Uncertainty-Aware Guided Volume Segmentation”. In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 1358–1365. DOI: 10.1109/TVCG.2010.208 (cit. on p. 7).

- [158]D. Pugmire, H. Childs, C. Garth, S. Ahern, and G. H. Weber. “Scalable Computation of Streamlines on Very Large Datasets”. In: *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. SC '09. Portland, Oregon: ACM, 2009, 16:1–16:12. DOI: 10.1145/1654059.1654076 (cit. on p. 73).
- [159]J. Raad and C. Cruz. “A Survey on Ontology Evaluation Methods”. In: *KEOD 2015 - Proceedings of the International Conference on Knowledge Engineering and Ontology Development, part of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2015), Volume 2, Lisbon, Portugal, November 12-14, 2015*. Ed. by A. L. N. Fred, J. L. G. Dietz, D. Aveiro, K. Liu, and J. Filipe. SciTePress, 2015, pp. 179–186. DOI: 10.5220/0005591001790186 (cit. on p. 140).
- [160]E. M. Reingold and J. S. Tilford. “Tidier Drawings of Trees”. In: *IEEE Trans. Software Eng.* 7.2 (1981), pp. 223–228. DOI: 10.1109/TSE.1981.234519 (cit. on p. 17).
- [161]B. Rieck, F. Sadlo, and H. Leitte. “Hierarchies and Ranks for Persistence Pairs”. In: *Topological Methods in Data Analysis and Visualization V: Theory, Algorithms, and Applications*. Feb. 2020 (cit. on p. 19).
- [162]A. Rind, M. Wagner, and W. Aigner. “Towards a Structural Framework for Explicit Domain Knowledge in Visual Analytics”. In: *2019 IEEE Workshop on Visual Analytics in Healthcare (VAHC)*. 2019, pp. 33–40. DOI: 10.1109/VAHC47919.2019.8945032 (cit. on pp. 129, 130, 132, 136).
- [163]M. Roy, S. Schmid, and G. Trédan. “Modeling and Measuring Graph Similarity: The Case for Centrality Distance”. In: *Proceedings of the 10th ACM International Workshop on Foundations of Mobile Computing*. FOMC '14. Association for Computing Machinery, 2014, pp. 47–52. DOI: 10.1145/2634274.2634277 (cit. on p. 60).
- [164]H. Saikia, H.-P. Seidel, and T. Weinkauff. “Extended Branch Decomposition Graphs: Structural Comparison of Scalar Data”. In: *Computer Graphics Forum* 33.3 (2014), pp. 41–50. DOI: 10.1111/cgf.12360 (cit. on pp. 18–20, 24).
- [165]J. Sanyal, S. Zhang, J. Dyer, A. Mercer, P. Amburn, and R. Moorhead. “Noodles: A tool for visualization of numerical weather model ensemble uncertainty”. In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 1421–1430 (cit. on pp. 8, 72).
- [166]D. Schneider, J. Fuhrmann, W. Reich, and G. Scheuermann. “A Variance Based FTLE-Like Method for Unsteady Uncertain Vector Fields”. In: *Topological Methods in Data Analysis and Visualization II: Theory, Algorithms, and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 255–268. DOI: 10.1007/978-3-642-23175-9_17 (cit. on p. 8).
- [167]D. Schneider, A. Wiebel, H. A. Carr, M. Hlawitschka, and G. Scheuermann. “Interactive Comparison of Scalar Fields Based on Largest Contours with Applications to Flow Visualization”. In: *IEEE Trans. Vis. Comput. Graph.* 14.6 (2008), pp. 1475–1482. DOI: 10.1109/TVCG.2008.143. URL: <https://doi.org/10.1109/TVCG.2008.143> (cit. on pp. 16, 18).

- [168]P. Schneider and K. Böttinger. “High-Performance Unsupervised Anomaly Detection for Cyber-Physical System Networks”. In: *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*. ACM. 2018, pp. 1–12. DOI: 10.1145/3264888.3264890 (cit. on p. 103).
- [169]A. Schnorr, D. N. Helmrich, D. Denker, T. W. Kuhlen, and B. Hentschel. “Feature Tracking by Two-Step Optimization”. In: *IEEE Transactions on Visualization and Computer Graphics* 26.6 (2020), pp. 2219–2233. DOI: 10.1109/TVCG.2018.2883630 (cit. on p. 54).
- [170]A. Schnorr, J. H. Göbbert, T. W. Kuhlen, and B. Hentschel. “Tracking space-filling structures in turbulent flows”. In: *5th IEEE Symposium on Large Data Analysis and Visualization, LDAV 2015, Chicago, IL, USA, October 25-26, 2015*. 2015, pp. 143–144. DOI: 10.1109/LDAV.2015.7348089 (cit. on p. 18).
- [171]C. Schulz, A. Nocaj, J. Goertler, O. Deussen, U. Brandes, and D. Weiskopf. “Probabilistic Graph Layout for Uncertain Network Visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (Jan. 2017), pp. 531–540. DOI: 10.1109/TVCG.2016.2598919 (cit. on p. 18).
- [172]S. V. Contest. *IEEE VIS 2016*. <http://www.uni-kl.de/sciviscontest/>. Accessed: 2019-11-28. 2016 (cit. on p. 47).
- [173]*IEEE VIS Scientific visualization contest 2017*. <https://www.dkrz.de/SciVis>. Accessed: 2021-03-03 (cit. on p. 66).
- [174]*Sea Ice Index Monthly Data by Year G02135 V3.0*. ftp://sidads.colorado.edu/DATASETS/NOAA/G02135/seaice_analysis/. 2020, Accessed: 2021-03-02 (cit. on p. 63).
- [175]L. Sevilla-Lara and E. Learned-Miller. “Distribution fields for tracking”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. June 2012, pp. 1910–1917. DOI: 10.1109/CVPR.2012.6247891 (cit. on p. 72).
- [176]B. A. Shapiro and K. Zhang. “Comparing multiple RNA secondary structures using tree comparisons”. In: *Computer Applications in the Biosciences* 6.4 (1990), pp. 309–318. DOI: 10.1093/bioinformatics/6.4.309 (cit. on p. 19).
- [177]Q. Shu, H. Guo, J. Liang, L. Che, J. Liu, and X. Yuan. “EnsembleGraph: Interactive visual analysis of spatiotemporal behaviors in ensemble simulation data”. In: *2016 IEEE Pacific Visualization Symposium (PacificVis)*. Apr. 2016, pp. 56–63. DOI: 10.1109/PACIFICVIS.2016.7465251 (cit. on p. 18).
- [178]V. Silva, E. Munch, and A. Patel. “Categorified Reeb Graphs”. In: *Discrete and Computational Geometry* 55 (Jan. 2015). DOI: 10.1007/s00454-016-9763-9 (cit. on p. 19).
- [179]C. Simon, P. Weber, and M. Sallak. *Data Uncertainty and Important Measures*. Jan. 2018. DOI: 10.1002/9781119489375 (cit. on p. 6).

- [180]G. Singh, T. Prabhakar, J. Chatterjee, V. Patil, S. Ninomiya, et al. “OntoViz: Visualizing Ontologies and Thesauri using Layout Algorithms”. In: *The Fifth International Conference of the Asian Federation for Information Technology in Agriculture (AFITA 2006)*. 2006 (cit. on p. 134).
- [181]T. Sobral, T. Galvão, and J. Borges. “An Ontology-Based Approach to Knowledge-Assisted Integration and Visualization of Urban Mobility Data”. In: *Expert Systems with Applications* 150 (2020), p. 113260. DOI: 10.1016/j.eswa.2020.113260 (cit. on pp. 126, 127).
- [182]B. -. Sohn and Chandrajit Bajaj. “Time-varying contour topology”. In: *IEEE Transactions on Visualization and Computer Graphics* 12.1 (2006), pp. 14–25. DOI: 10.1109/TVCG.2006.16 (cit. on pp. 53, 54).
- [183]R. Sridharamurthy, T. Bin Masood, A. Kamakshidasan, and V. Natarajan. “Edit Distance between Merge Trees”. In: *IEEE Transactions on Visualization and Computer Graphics* (2018), pp. 1–1. DOI: 10.1109/TVCG.2018.2873612 (cit. on p. 19).
- [184]H. Stitz, S. Gratzl, H. Piringer, T. Zichner, and M. Streit. “KnowledgePearls: Provenance-Based Visualization Retrieval”. In: *IEEE Transactions on Visualization and Computer Graphics (VAST '18)* 25.1 (2018), pp. 120–130. DOI: 10.1109/TVCG.2018.2865024 (cit. on p. 126).
- [185]F. Stoffel, F. Fischer, and D. A. Keim. “Finding Anomalies in Time-series Using Visual Correlation for Interactive Root Cause Analysis”. In: *Proceedings of the Tenth Workshop on Visualization for Cyber Security. VizSec '13*. Atlanta, Georgia, USA: ACM, 2013, pp. 65–72. DOI: 10.1145/2517957.2517966. URL: <http://doi.acm.org/10.1145/2517957.2517966> (cit. on p. 103).
- [186]M.-A. Storey, M. Musen, J. Silva, C. Best, N. Ernst, R. Ferguson, and N. Noy. “Jambalaya: Interactive Visualization to Enhance Ontology Authoring and Knowledge Acquisition in Protégé”. In: *Workshop on Interactive Tools for Knowledge Capture*. Vol. 73. Dec. 2001 (cit. on p. 134).
- [187]*Study on Communication for Automation in Vertical Domains*. 3GPP TR 22.804, V1.0.0. 2017 (cit. on p. 102).
- [188]J. Sweller. “Element interactivity and intrinsic, extraneous, and germane cognitive load”. In: *Educational psychology review* 22.2 (2010), pp. 123–138. DOI: 10.1007/s10648-010-9128-5 (cit. on p. 171).
- [189]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)*. 2005, pp. 136–144. DOI: 10.1109/SMI.2005.45 (cit. on p. 53).
- [190]A. Szymczak. “Morse connection graphs for piecewise constant vector fields on surfaces”. In: *Computer Aided Geometric Design* 30.6 (2013), pp. 529–541 (cit. on pp. 74, 96).
- [191]K.-C. Tai. “The Tree-to-Tree Correction Problem”. In: *J. ACM* 26.3 (1979), pp. 422–433. DOI: 10.1145/322139.322143 (cit. on pp. 19, 21).

- [192]H. Theisel, T. Weinkauff, H.-C. Hege, and H.-P. Seidel. “Topological Methods for 2D Time-Dependent Vector Fields Based on Stream Lines and Path Lines”. In: *IEEE Trans. Vis. Comput. Graph.* 11.4 (2005), pp. 383–394. DOI: 10.1109/TVCG.2005.68. URL: <https://doi.org/10.1109/TVCG.2005.68> (cit. on p. 53).
- [193]D. M. Thomas and V. Natarajan. “Symmetry in Scalar Field Topology”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (Dec. 2011), pp. 2035–2044. DOI: 10.1109/TVCG.2011.236 (cit. on p. 18).
- [194]J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. “The Topology ToolKit”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (Jan. 2018), pp. 832–842. DOI: 10.1109/TVCG.2017.2743938 (cit. on pp. 38, 55).
- [195]C. Tominski, P. Schulze-Wollgast, and H. Schumann. “3D information visualization for time dependent data on maps”. In: *Ninth International Conference on Information Visualisation (IV’05)*. July 2005, pp. 175–181. DOI: 10.1109/IV.2005.3 (cit. on p. 103).
- [196]X. Tricoche, T. Wischgoll, G. Scheuermann, and H. Hagen. “Topology tracking for the visualization of time-dependent two-dimensional flows”. In: *Comput. Graph.* 26.2 (2002), pp. 249–257. DOI: 10.1016/S0097-8493(02)00056-0. URL: [https://doi.org/10.1016/S0097-8493\(02\)00056-0](https://doi.org/10.1016/S0097-8493(02)00056-0) (cit. on p. 53).
- [197]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, Í. 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”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2 (cit. on p. 153).
- [198]Vivek Verma and A. Pang. “Comparative flow visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* 10.6 (2004), pp. 609–624. DOI: 10.1109/TVCG.2004.39 (cit. on p. 133).
- [199]L. von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, J. Pfrommer, A. Pick, R. Ramamurthy, M. Walczak, J. Garcke, C. Bauckhage, and J. Schuecker. “Informed Machine Learning – A Taxonomy and Survey of Integrating Knowledge into Learning Systems”. In: *arXiv e-prints*, arXiv:1903.12394 (Mar. 2019), arXiv:1903.12394. arXiv: 1903.12394 [stat.ML] (cit. on p. 149).
- [200]M. Wagner, D. Slijepcevic, B. Horsak, A. Rind, M. Zeppelzauer, and W. Aigner. “KAVAGait: Knowledge-Assisted Visual Analytics for Clinical Gait Analysis”. In: *IEEE Transactions on Visualization and Computer Graphics* 25.3 (2019), pp. 1528–1542. DOI: 10.1109/TVCG.2017.2785271 (cit. on p. 136).
- [201]C. Wang and J. Tao. “Graphs in Scientific Visualization: A Survey”. In: *Computer Graphics Forum* 36.1 (2017), pp. 263–287. DOI: 10.1111/cgf.12800 (cit. on p. 75).

- [202]C. Wang, H. Yu, and K.-L. Ma. “Application-driven compression for visualizing large-scale time-varying data”. In: *IEEE Computer Graphics and Applications* 30.1 (2010), pp. 59–69 (cit. on p. 75).
- [203]J. Wang, S. Hazarika, C. Li, and H. Shen. “Visualization and Visual Analysis of Ensemble Data: A Survey”. In: *IEEE Transactions on Visualization and Computer Graphics* 25.9 (Sept. 2019), pp. 2853–2872. DOI: 10.1109/TVCG.2018.2853721 (cit. on p. 18).
- [204]L. Wang and T. Jiang. “On the Complexity of Multiple Sequence Alignment”. In: *Journal of Computational Biology* 1.4 (1994), pp. 337–348. DOI: 10.1089/cmb.1994.1.337 (cit. on p. 23).
- [205]X. Wang, D. H. Jeong, W. Dou, S.-W. Lee, W. Ribarsky, and R. Chang. “Defining and applying knowledge conversion processes to a visual analytics system”. In: *Computers & Graphics* 33.5 (2009), pp. 616–623. DOI: 10.1016/j.cag.2009.06.004 (cit. on p. 124).
- [206]G. H. Weber, S. E. Dillard, H. Carr, V. Pascucci, and B. Hamann. “Topology-Controlled Volume Rendering”. In: *IEEE Transactions on Visualization and Computer Graphics* 13.2 (Mar. 2007), pp. 330–341. DOI: 10.1109/TVCG.2007.47 (cit. on p. 13).
- [207]G. H. Weber, S. E. Dillard, H. Carr, V. Pascucci, and B. Hamann. “Topology-Controlled Volume Rendering”. In: *IEEE Transactions on Visualization and Computer Graphics* 13.2 (Mar. 2007), pp. 330–341. DOI: 10.1109/TVCG.2007.47 (cit. on p. 16).
- [208]M. Weber, M. Alexa, and W. Müller. “Visualizing time-series on spirals”. In: *Symposium on Information Visualization (INFOVIS)*. Vol. 1. 2001, pp. 7–14. DOI: 10.1109/INFVIS.2001.963273 (cit. on pp. 103, 110).
- [209]R. T. Whitaker, M. Mirzargar, and R. M. Kirby. “Contour boxplots: A method for characterizing uncertainty in feature sets from simulation ensembles”. In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), pp. 2713–2722 (cit. on pp. 8, 72).
- [210]B. Whitlock, J. M. Favre, and J. S. Meredith. “Parallel in Situ Coupling of Simulation with a Fully Featured Visualization System”. In: *Proceedings of the 11th Eurographics Conference on Parallel Graphics and Visualization*. EGPGV ’11. Llandudno, UK: Eurographics Association, 2011, pp. 101–109. DOI: 10.2312/EGPGV/EGPGV11/101-109 (cit. on p. 75).
- [211]W. Widanagamaachchi, C. Christensen, V. Pascucci, and P. Bremer. “Interactive exploration of large-scale time-varying data using dynamic tracking graphs”. In: *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. 2012, pp. 9–17. DOI: 10.1109/LDAV.2012.6378962 (cit. on p. 54).
- [212]V. Wijk. “The Value of Visualization”. In: *VIS 05. IEEE Visualization, 2005*. 2005, pp. 79–86. DOI: 10.1109/VISUAL.2005.1532781 (cit. on p. 124).
- [213]J. Woodring, J. Ahrens, J. Figg, J. Wendelberger, S. Habib, and K. Heitmann. “In-situ Sampling of a Large-Scale Particle Simulation for Interactive Visualization and Analysis”. In: *Computer Graphics Forum*. Vol. 30. 3. Wiley Online Library, 2011, pp. 1151–1160 (cit. on p. 75).

- [214]K. Wu and S. Zhang. “A contour tree based visualization for exploring data with uncertainty”. In: *International Journal for Uncertainty Quantification* 3 (Jan. 2013), pp. 203–223. DOI: 10.1615/Int.J.UncertaintyQuantification.2012003956 (cit. on p. 19).
- [215]L. Xu and H.-W. Shen. “Flow web: A graph based user interface for 3D flow field exploration”. In: *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics. 2010, 75300F–75300F (cit. on pp. 73, 77–79).
- [216]L. Yan, Y. Wang, E. Munch, E. Gasparovic, and B. Wang. “A Structural Average of Labeled Merge Trees for Uncertainty Visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* (2019), pp. 1–1. DOI: 10.1109/TVCG.2019.2934242 (cit. on p. 19).
- [217]C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh. “Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets”. In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. Dec. 2016, pp. 1317–1322. DOI: 10.1109/ICDM.2016.0179 (cit. on pp. 105, 148).
- [218]H. Yu, C. Wang, R. W. Grout, J. H. Chen, and K.-L. Ma. “In situ visualization for large-scale combustion simulations”. In: *IEEE computer graphics and applications* 30.3 (2010), pp. 45–57 (cit. on p. 75).
- [219]B. Zehner, N. Watanabe, and O. Kolditz. “Visualization of gridded scalar data with uncertainty in geosciences”. In: *Computers & Geosciences* 36.10 (2010), pp. 1268–1275. DOI: 10.1016/j.cageo.2010.02.010 (cit. on p. 8).
- [220]C. Zhang, T. Schultz, K. Lawonn, E. Eisemann, and A. Vilanova. “Glyph-Based Comparative Visualization for Diffusion Tensor Fields”. In: *IEEE Transactions on Visualization and Computer Graphics* 22.1 (2016), pp. 797–806. DOI: 10.1109/TVCG.2015.2467435 (cit. on p. 133).
- [221]K. Zhang. “A Constrained Edit Distance Between Unordered Labeled Trees”. In: *Algorithmica* 15.3 (1996), pp. 205–222. DOI: 10.1007/BF01975866 (cit. on p. 19).

List of Figures

1.1	xkcd-errorbars	2
3.1	Fuzzy Contour Trees	13
3.2	Three-dimensional scalar field rendering	15
3.3	Contour Tree	16
3.4	Branch Decomposition and Layout Options	17
3.5	Differences Between Alignment and Edit Distance	21
3.6	Consistent Root and Path Properties	24
3.7	Inner Node Turned into a Leaf	26
3.8	The Fuzzy Contour Tree Layout	29
3.9	Branch Spacing Optimization	31
3.10	The Fuzzy Contour Tree User Interface	32
3.11	Branch Highlighting	33
3.12	Tree and Component Highlighting	34
3.13	Different Layouts and Weights	35
3.14	Challenge in the Bundled Layout	35
3.15	Different Challenges and their Solutions	36
3.16	The Outlier Ensemble	38
3.17	Fuzzy Contour Tree of the Outlier Ensemble	39
3.18	The Differing Structure of the Outlier	39
3.19	Member Highlighting	40
3.20	Component Highlighting	40
3.21	The Scattered Peaks Ensemble	41
3.22	The Fuzzy Contour Tree of the Scattered Peaks Ensemble	42
3.23	Member Highlighting in the Scattered Peaks Ensemble	43
3.24	The Convection Simulation	43
3.25	Component Highlighting in the Convection Simulation	44
3.26	Tree Highlighting in the Convection Simulation	45
3.27	Common Topological Structures of the Convection Simulation	46
3.28	Separation of Members in the Convection Simulation	46
3.29	Matching of Unrelated Branches	47

4.1	The Time-Varying Fuzzy Contour Tree Interface	52
4.2	Scalar Value Choices in the Alignment	54
4.3	The Trickle-Down-Layout of the Time-Varying Fuzzy Contour Tree	56
4.4	Horizontal Positioning	57
4.5	Two-Colored Main Branch	58
4.6	The Time Selector	59
4.7	Coloring Based on Sub-Alignments	59
4.8	Time Selector Options	60
4.9	Tree Highlighting	62
4.10	Annual Sea Ice Extent	63
4.11	Sea Ice Extent in January and August	63
4.12	Insights using Periodic Selection	64
4.13	Sea Ice Extent During Individual Months	65
4.14	Sliding Window	67
4.15	Limitations of Time-Varying Fuzzy Contour Trees	68
5.1	Visitation Maps	69
5.2	The Euler Method	71
5.3	Visitation Map Generation From a Flow Graph	74
5.4	Ad Hoc Visitation Map Creation vs. Using Visitation Graphs	76
5.5	Visitation Graph Creation	78
5.6	Probability Approximation	81
5.7	Comparison of Traditional and Approximated Visitation Map	82
5.8	Visitation Map Approximation	83
5.9	Probabilities to be Hit per Time Step	86
5.10	Different Grid Resolutions	86
5.11	Glyph Visualization	87
5.12	Visitation Graph Tradeoffs	88
5.13	Dependency on Member Count	89
5.14	Creation Times of Visitation Maps Starting in Multiple Cells	90
5.15	Visitation Graphs as an Alternative to Downsampling	91
5.16	Industrial Stirring Simulation	92
5.17	Exploration of the Convection Simulation	94
5.18	The Cavity Flow Simulation	95
6.1	The Security in Process System	101
6.2	Anomaly Detection Results	107
6.3	The Security in Process GUI	108
6.4	Spiral Plot Features	109
6.5	Anomaly Detection Results Extended	114

6.6	Sub-Processes	114
6.7	Anomaly #1	116
6.8	Anomaly #2	116
6.9	Cycle Length Adjustment	117
6.10	Anomaly #3	117
6.11	Anomaly #4	118
6.12	Anomaly #5	119
7.1	The Knowledge Rocks Framework	122
7.2	The KAVA Model	124
7.3	Data Flow	130
7.4	Acting Ontology	131
7.5	Acting Ontology For Knowledge-Assisted Comparative Assessment of Breast Cancer	136
7.6	Acting Ontology For Draco	137
8.1	The Knowledge-Assisted Security in Process System	142
8.2	Database Schema	144
8.3	The Incident Based Acting Ontology	145
8.5	Training Data	150
8.6	Workflow in the enhanced Security in Process System	152
8.7	Previous and Enhanced Time Slider	153
8.8	Original and Enhanced Spiral Plot	154
8.9	Overview Using the Time Slider	156
9.1	The Original Time Slider	161
9.2	Readings Rated as Alert	161
9.3	Periods	162
9.4	Irregularities	162
9.5	Different Incident Types	163
9.6	Evaluation Results for Effectiveness	165
9.7	Evaluation Results for Satisfaction	168

CV - Anna-Pia Lohfink

School and Professional Training.

- 8/99- Hildegardisschule Bingen
- 3/08 Graduation: Abitur (german university entrance qualification)
- 4/08- Up to data professional services GmbH Wörrstadt
- 3/10 Professional education
Graduation: Computer Scientist in Application Software Development

Studies

- 9/12 Technische Universität Kaiserslautern
Bachelor in Mathematics, minor field of study electrical engineering
Area of specialization: Algebra, number theory, cryptography
[Looking at the Cryptophon from the Cryptographic Point of View – the IDEA Cryptosystem](#)
- 8/15 Master in Mathematics, minor field of study Computer Science,
Area of specialization: Cryptography, computer graphics and visualization
[Cryptography in braid groups](#)

Internships and Experiences Abroad.

- 3-5/12 12 weeks internship at the IT department of Cognizant, India in Pune
- 10/12-2/13 Semester abroad in Strasbourg, France, field of study: Mathematics, Logic
- 2-4/14 7 weeks internship at CERN joining the Root developer team for visualization
- 8-12/19 Research stay at Lawrence Berkeley National Laboratory, California.
Research topic: Fuzzy Contour Trees.

To Dummy.jpg – you were the first figure in all of my papers, but you never made it to the deadline. Now it's your time:

