Visual Analysis of Multi-run Spatio-temporal Simulation Data

BY ALEXEY FOFOV

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in Computer Science

Dissertation Committee:

Prof. Dr. Lars Linsen
Prof. Dr. Peter Baumann
Prof. Dr. Rüdiger Westermann

Statutory Declaration
(on Authorship of a Dissertation)

I, Alexey Fofonov, hereby declare that I have written this PhD thesis independently, unless where clearly stated otherwise. I have used only the sources, the data and the support that I have clearly mentioned. This PhD thesis has not been submitted for conferral of degree elsewhere.

I confirm that no rights of third parties will be infringed by the publication of this thesis.

Bremen, February 9, 2016

Signature
Abstract

Multi-run simulations are widely used to investigate how the simulated processes evolve depending on varying initial conditions. This kind of experiments is typically executed using high-performance computers and usually generates a large amount of data to analyze. Multi-run simulations are very useful for different fields of sciences, such as astrophysics, ballistics, volcanology, and oceanography. Therefore, a lot of data needs to be analyzed. Visual representation and analysis of multi-run multi-variate time-varying volume data require fast and robust computational algorithms as well as novel visualization approaches.

The main goals of this work are an efficient visualization of multi-run time-varying spatial data, pairwise and ensemble-wise comparison of simulations, as well as simulation feature detection and investigation, and exploration and classification of simulation results. Besides that, the current work aims generalization of developing approaches to be applicable for data of any spatial configuration and of a multi-field nature. Such task consists of investigation and developing visualization methods to be applied on different aggregation levels from a field distribution of an single time frame up to overview of the whole simulations ensemble.

This work introduces a number of approaches and tools, which can significantly improve an analysis of multi-run time-varying spatial data on all aggregation levels. To investigate a single-field spatial data a new isocontour based similarity measure is introduced. It allows for a fast and robust isosurface similarity maps extraction to present a structural information of a volume data, and also for a qualitative comparison of different data fields. Using a quasi-Monte Carlo method the measure is independent on a spatial structure of data and very efficient in terms of computational costs. Taking into account a geometry of isocontours the measure captures much more information comparing to statistical descriptors. Further generalization of the measure allows to compare entire data fields including a multi-field information.

For an interactive visualization and for detailed comparative analysis of multi-field spatial data algorithms for a splat-based progressive rendering of continuous scatterplots are developed. Continuous scatterplots are used to investigate a data structure of an individual frame of multivariate data defined on a continuous domain. Designed in a GPU parallel fashion it boosts up the rendering for hundreds times comparing to other techniques for both linear and logarithmic representation scales. Calculation of similarity between multiple fields is also possible using the introduced measure.

For a comparative visualization of multi-run and multi-field spatio-temporal simulation data a new visualization technique is introduced. Using the proposed measure it is possible to compute a similarity matrix of all the time frames of all the simulations in an analyzed ensemble. Then, using multidimensional scaling (MDS) projection
method the matrix can be projected to a low-dimensional space. The main idea is to represent each simulation as a polyline on a shared two- or three-dimensional output of successively connected points, which are positions of the corresponding simulation time frames relatively to all other simulation time frames in the ensemble. The distances on the projection reflect the distances from the similarity matrix. This allows to visually recognize behavior patterns of simulations (or patterns of multiple data fields of a single simulation) and their outliers, as well as other data features. Another application of the technique is estimation of influence of simulation parameters and initial conditions to the simulation behavior. Applying this approach to real world examples from different research fields, namely climate and astrophysical simulations, and of different spatial domain configuration, Eulerian and Lagrangian systems correspondingly, its usefulness is discussed and confirmed by domain experts.

For an overview of a multi-run and multi-field spatio-temporal simulation data on a highest aggregation levels two visualizations are proposed. The goal of both visualizations is to narrow down an area of multi-run data analysis by early detection of regions of interests in terms of data field and time ranges. The field distribution histogram aggregates information about all the time frames and all the simulations within an ensemble. It shows a probability of occurrence of field values over the entire ensemble and allows to identify simulations with outstanding values, as well as estimate field range of interest for a further analysis. The function plot aggregates information about the time frames for each simulation individually. The plot represents the function values of each spatial data sample of each simulation run as a piece-wise linear graph of a time series. Then for the visual encoding, the time series lines aggregate over a 2D grid leading to a 2D density histogram and a transfer function can be applied to map the accumulated density values to color. Such representation allows to distinguish and compare field distribution patterns for all the simulations in ensemble side by side or accumulating. With introduced interaction tools it is possible to define regions of interest, behavior features and align simulations by time if desired. Moreover, interactive selections with a further progressive rendering of point graphs are introduced. It allows to follow patterns on the plot and show corresponding point graphs for other data fields, discovering correlations and differences in their behavior.

Finally, a scheme of a top-down analysis of a multi-run spatio-temporal simulation data is introduced. In this work an application on the real world data sets, as well as detailed discussions about scalability of the methods, their efficiency and restrictions, are presented. All the introduced aggregated visualizations can be linked to a domain visualization making the interactive visual analysis efficient and intuitive, which is confirmed by performed user study and domain experts.
## Contents

1 Introduction ................................................. 1

2 Related work ............................................. 7

3 Isocontour similarity measure .......................... 15
   3.1 Introduction ........................................ 16
   3.2 Related Work ...................................... 17
   3.3 Distance Computation ............................... 17
      3.3.1 Mutual Information ......................... 17
      3.3.2 Quasi-Monte Carlo Approach ............... 19
      3.3.3 Sensitivity of the Measures ............... 20
   3.4 Implementation ................................... 22
   3.5 Results and Discussion ........................... 23
      3.5.1 Similarity Map Results .................... 23
      3.5.2 Computation Times ......................... 25
   3.6 Conclusion ......................................... 26

4 Progressive rendering of continuous scatterplots . 27
   4.1 Introduction ....................................... 28
   4.2 Related Work ..................................... 30
   4.3 Background ........................................ 32
      4.3.1 Basic Terms and Notations ............... 32
      4.3.2 Direct Approach for Generation of Continuous Scatterplots . 34
      4.3.3 Progressive Approach ...................... 35
      4.3.4 Spatially Unstructured Data ............... 38
   4.4 Spectral Representation of Continuous Scatterplots . 38
   4.5 Progressive Algorithm ............................ 40
   4.6 Hybrid Algorithm ................................ 41
   4.7 Implementation ................................... 42
   4.8 Results ............................................ 43
   4.9 Conclusion ......................................... 49
CONTENTS

5 Continuous representation of projected attribute spaces 51
  5.1 Introduction ................................................. 52
  5.2 Background .................................................. 55
    5.2.1 Physical, Attribute, and Visual Domains .............. 55
    5.2.2 Spatially Unstructured Data ............................ 57
  5.3 CoRPAS ....................................................... 58
    5.3.1 User-defined Projection ................................. 60
    5.3.2 Kernel Functions and Their Supports .................. 62
    5.3.3 Footprint Computation .................................. 63
    5.3.4 Construction of Logarithmic Plots ..................... 66
  5.4 Results ...................................................... 67
  5.5 Conclusion ................................................... 72

6 Multi-run similarity plots 73
  6.1 Introduction ................................................ 74
  6.2 Multi-run Plot .............................................. 76
    6.2.1 Distance Function ...................................... 76
    6.2.2 Projection Algorithm .................................... 77
  6.3 Scalability of Implementation ................................ 78
  6.4 Visual Analysis Based on Multi-run Plot ..................... 81
  6.5 Application to Gridded Data Analysis ......................... 86
  6.6 Application to Unstructured Data Analysis .................... 89
  6.7 Parameter Space Visualization ................................ 95
  6.8 Distance Function Analysis .................................. 99
  6.9 User Study of Visual Analysis ............................... 102
  6.10 Discussion .................................................. 105
    6.10.1 Quasi-Monte Carlo Sampling .............................. 105
    6.10.2 Temporal Subsampling .................................. 106
    6.10.3 Dimensionality ......................................... 107
    6.10.4 Investigating Projection Dimensions ................... 108
    6.10.5 Robustness of Isosurface Similarity Plots ............ 110
    6.10.6 Computation Times ...................................... 110
    6.10.7 Sensitivity of the Approach ............................. 111
    6.10.8 Visual Complexity ...................................... 113
  6.11 Conclusion .................................................. 116

7 MultiVisA tool and application scenarios 117
  7.1 Introduction ................................................ 118
  7.2 Visual Analysis of Multi-run Simulation Data ............... 120
    7.2.1 Field Distribution Histogram ............................ 122
    7.2.2 Function Plot ............................................ 123
Chapter 1

Introduction

Numerical modeling is a one of the most developing and important parts of a modern research. The modeling of physical phenomena requires to perform many time-varying multi-variate simulations over spatial domains and structures. Such simulations in hydrodynamics, astrophysics, climatology, engineering and other fields produce data of permanently increasing size. Commonly, simulation data consist of thousands of time steps and describe evolution of tens of scalar and vector fields. To investigate the dependence of the process on initial conditions, hundreds of simulations have to be performed with varying parameters values and initial conditions. The resulting ensembles of multi-run simulation data have therefore both: large size and complicated structure.

A formal definition of the multi-run multi-field simulation data ensemble $E$, consisting of $s$ simulation runs $S_i$ with $m$ scalar fields $f_{ij}$ (vector fields can be also considered as a combination of scalar fields), can be summarized as follows: $E = \{S_i, i = 1..s\}$, where $S_i = \{f_{ij}(x,t) : (\mathbb{R}^n, N) \to \mathbb{R}, j = 1..m, x \in \mathbb{R}^n, t = 0..N_i\}$. Usually, physically based simulations are varying over time $t$ by $N_i$ discretized equidistantly or non-equidistantly time intervals (for each $S_i$ the number of time intervals $N_i$ can be different) and performed in a two- or three-dimensional space ($n = 2$ and $n = 3$ correspondingly). The differences between $S_i$ are derived by initial parameters (categorical or numerical) or initial conditions. Analysis of such data aims for different global tasks:
• Estimation of the parameters’ influence on the simulations’ results and behavior.
• Comparison of the simulations’ results and behavior.
• Identification of simulations’ features, behavior patterns and outliers.

Each of those global tasks consists of many smaller tasks on every data aggregation level from the highest one (analysis of a whole simulation ensemble) to the lowest one (analysis of a single scalar field of a single time frame of a single run). The general task on all data aggregation levels is to narrow down an analysis area until the desired information can be extracted.

Analysis and processing of ensemble data require methods for a qualitative comparison and visual exploration of simulation results. The huge size and high complexity of data cause numerous difficulties when operating. Due to multidimensional nature of such data, one needs to take into account many attributes and features describing each data frame in order to perform an efficient and valuable analysis. Manual exploration over time and space is too time consuming and inefficient, while statistical descriptors are too general and do not capture individual details. On the other hand, existing tools for visualization and analysis of multi-run time-varying multi-variate simulations are too specific in their application areas, forcing researchers to implement their own sub-routines. The data can be either volumetric (spatial relations of samples are given) or of general structure. It can affect the choice of appropriate techniques.

The aim of this research is to develop visualization techniques and tools for scientists, who are doing such multi-run simulation analysis. Simulation analysis is important not less than the simulation itself, and with appropriate tools it could be done fast and efficiently, saving time and resources. Proper visualizations of data and its features, behavior patterns of simulations and their outliers, supported by interactive tools and methods bring the analysis process to a high level of efficiency and allow for a better understanding of the simulated phenomena.

The main goal of our research is to develop a novel comprehensive system for visual representation and analysis of multi-run multi-field data. The proposed system consists of several stages (each is described in its chapter) aiming at different detalization level of processed information. Throughout the work we present approaches and techniques
for the lowest level of the data analysis (single scalar field over spatial domain), intermediate level (multi-field spatial data, single-run time-variante spatial simulation data, ensemble of spatial data fields for a certain time frame), and up to the highest level (ensemble of multi-run multi-field time-variante spatial simulations).

Any spatio-temporal simulation consists of many individual time steps, therefore it is important to have a proper descriptor of these units. Such proper descriptor has to allow to estimate relations between time steps for a further construction of entire picture of simulation behavior. Therefore, in Chapter 3 we provide an analysis of a single field spatial data. Isosurfaces play a crucial role in visualization and interpretation of volumetric data. We introduce a novel isosurface similarity measure which allows to investigate a spatial data structure by comparing a set of isosurfaces, sampling the analyzed field range and computing isosurface similarity maps. We show the efficiency of the measure by applying it to different data samples and comparing to the state of the art technique. In our examples we show that our approach can be applied to data of any spatial configuration; it is up to 130 times faster and it is less sensitive to the spatial distribution of (nonoverlapping) features and rather captures their shape differences.

Commonly spatio-temporal simulations describe phenomena with multivariate nature. Therefore, it is of interest to investigate a multivariate data structure. Chapter 4 describes a way to analyze visually pairwise relations between different scalar fields of spatial data. Continuous scatterplots are widely used to investigate such relations within continuous spatial domains. We introduce an algorithm for a fast GPU based progressive rendering of continuous scatterplots for unstructured point-based spatial data, which allows for an interactive high resolution rendering. Applying this technique to real world data, we show the effectiveness of the continuous scatterplots for point-based astrophysical simulations and compare the effectiveness of the rendering algorithm with the state of the art approach; in particular, the speed-up of 500 times is achieved in our tests.

However, a number of data attributes can be very large and in case their importance has to be estimated the described pairwise analysis would be a tedious task. Further we investigate an analysis of a multi-field spatial data using continuous representation of projected attribute spaces, described in Chapter 5. Similarly to representation of pairwise relations between scalar fields using continuous scatterplots, we present an
extension of such representation for multi-variate cases. Besides that we show that dif-
ferent scales of representation axes can be valuable and describe an algorithm for a fast
rendering of continuous scatterplots in the case of a logarithmic scale. Our algorithm
is the first one, which allows to visualize continuous scatterplots for both gridded and
unstructured (e.g., smoothed particles hydrodynamics) volumetric data in interactive
fashion, exploiting the continuity property of data.

In Chapter 6 we present a novel approach for a visual analysis of multi-run spatio-
temporal simulation data: multi-run similarity plots. Using the proposed isosurface
similarity measure we compute distances between all the simulations’ time steps within
an analyzed ensemble. Having all such distances, we construct a distance matrix and
project this information to a low dimensional space for a further visualization, where
every simulation is presented as a polyline of successively connected points (which are
the time steps). Such representation allows to identify and compare behavior patterns
and data features visually. We show that in conjunction with appropriate tools and inter-
action techniques, such as linked views to a physical domain, color coding, interactive
selections and projection recalculations and others, one can successfully analyze real
world data sets from different application areas and of different spatial structures. We
also present results of a performed user study of the approach, results of comparison
with an existing technique and evaluation from domain experts. We show that the pro-
posed method allows to effectively capture detailed information about simulation runs
basing on a shape of analyzed data field, instead of just using means or other statisti-
cal descriptors as the other methods do. The proposed data visualization using multi-
dimensional scaling projection allows for an intuitive visual perception of main data
features, such as behavior patterns, and for a visual comparison in terms of its prin-
cipal components. These techniques constitute our contributions in the visual encoding
of multi-run spatio-temporal data allowing for the comprehension of the entire set of
simulation runs and an interactive visual analysis system based on coordinated views
between physical domain, parameter and projection spaces.

For a fruitful and efficient analysis of the multi-run multi-field spatio-temporal sim-
ulation data, it is necessary to be able to narrow down the area of interest to be consid-
ered. We propose a top-down approach, which allows to investigate data features from
early on of the analysis process. Using different data visualization methods on differ-
ent aggregation levels, we allow to identify data features and investigate simulations’
behavior patterns drilling down from the first general overview of an entire ensemble
until a detailed comparison of individual time steps. Starting with a field distribution
histograms, we aggregate information about scalar fields over all the simulation runs
and all the time steps to see the structure of field values occurrences. Using this infor-
mation, we define regions of interest for a further analysis, detecting simulations with
outstanding field values and restricting an analyzed field range. Then we create function
plots which show an aggregated information over simulations but not over time. Col-
lecting information about function values of each spatial data sample as a piece-wise
linear graph of a time series, we build a 2D histogram for each analyzed simulation.
Applying a transfer function with an adjustable color scheme to the histograms, we can
highlight data features, behavior patterns and a data structure over time. Thus, we are
able to perform visual comparisons of different simulations, as well as compare differ-
ent data fields. Moreover, we introduce an approach for interactive selections of points
on the plots with a further progressive rendering of their trajectories (i.e., connected
function values). One can also show selected points in a physical domain to investigate
the structures recognized on the plots. The function plots also allow to select time in-
tervals of interest for a further analysis or align simulations by a desired feature. This
visualization allows to narrow down the amount of analyzed data and therefore to aim
more precisely analyzed data features for other analyses approaches. Then we apply our
multi-run similarity plots for a detailed comparative analysis of individual time steps.
We show that using results of the analysis on highest aggregation levels one can sig-
nificantly improve effectiveness and quality of the similarity plots, as well as support
other existing methods. Finally, we developed an application for a complex visual anal-
ysis of multi-run multi-field spatio-temporal simulation data called ”MultiVisA”, which
includes all three described techniques designed to support each other in interactive
fashion. The proposed approaches and the ”MultiVisA” application are presented in
Chapter 7 with corresponding discussions about their applicability, scalability and ap-
plication to real world examples with a feedback from domain experts. The proposed
top-down approach forms a general pipeline for the multi-run simulations data analysis,
providing an efficient way to combine the proposed techniques and tools in order to in-
vestigate data features. It allows researchers to find and locate desired features easily in
an interactive visual fashion within a whole ensemble, and to focus on them in terms of a
certain time interval and field range independently on the data structure and application area, when normally these tasks require from researchers to write specific subroutines and spend days or even weeks to process the whole amount of data.

Despite of a successful application of the isosurface based measure for generation of multi-run similarity plots, such approach has some limitations and requirements due to its nature (e.g., one need to choose a representative isovalue for the analysis). In Chapter 8 we generalize the proposed measure to allow for a comparison of entire fields. Moreover, we extend this technique to capture a multi-field data structure in a single descriptor. Thus, it becomes possible to perform a multi-run multi-field spatio-temporal data analysis without any preprocessing steps for entire ensembles taking into account all the available information. We show different ways for visualization of projected information and discuss their applications and limitations, as well as compare our measure with other existing measures based on field gradients’ information and correlation coefficients.

The proposed visualization pipeline for a multi-run multi-field spatio-temporal simulation data is tested with several real world data sets, provided us directly from different application areas, and evaluated by corresponding domain experts. The presented results are assessed in terms of quality, scalability and computational speed. The developed techniques were implemented in a GPU parallel fashion for a better performance and compared to similar approaches where possible.
Chapter 2

Related work

The problem of visualization and analysis of multi-run time-varying simulation data is critical for many sciences. Thus, many problems associated with data size, visual analysis and comparative interactive representation exist. The data can be either volumetric (spatial relations of samples are given) or of general structure. It can affect the choice of appropriate techniques. A large number of researchers deal with the visualization and analysis of time-dependent and multi-variate data, e.g., Aigner et al. [1] develop and discuss a systematic view on the diversity of methods for visualizing time-oriented data, Fuchs and Hauser [2] discuss how different techniques take effect at specific stages of the visualization pipeline and how they apply to multi-variate data sets being composed of scalars, vectors and tensors. High-dimensional data visualization techniques were reviewed by Grinstein et al. [3]. The authors presented most useful state-of-the-art approaches to visualize such data, illustrating their advantages and limitations. Some problems of visual analysis of multi-dimensional medical data are highlighted by Bernatavičienė et al. [4]. In their survey, Kehrer and Hauser [5] study existing methods for visualization and interactive visual analysis of multi-faceted scientific data, which include multi-run data as a particular example. Thus, we can distinguish four main directions of the analysis, which belong to the problem.
Analysis of scalar fields

For a visual analysis of single variate data, in particular volumetric scalar fields, many simple visualization approaches are widely used: histograms, plots, diagrams and others. Most of them collect some statistics about the data field. For example, continuous histograms, as an 1-D version of continuous scatterplots, can be used to examine isosurface statistics for analyzing 3-D scalar fields (Carr et al. [6] and Scheidegger et al. [7]). For a detailed comparison of different scalar fields, structural information about field distributions can be used, such as isocontours and isosurfaces. Schneider et al. [8] propose to compare scalar fields basing on data features, which are defined by largest contour segmentation after topological simplification. Relationships defined by the similarity measure based on contours overlapping are ranked and presented in a thumbnail gallery of feature pairs and a graph representation showing all relationships between individual contours. However, the authors conclude that it is not straightforward to extend their work to time-varying data. The concept of isosurface similarity maps is introduced by Bruckner and Möller [9]. They present structural information of a volume data set by depicting similarities between individual isosurfaces quantified by a robust information-theoretic measure. One obvious disadvantage of the approach is the considerable cost of generating the isosurface similarity map. Given implementation can require several hours of processing time for a single data frame. Haidacher et al. [10] introduce multimodal surface similarity maps as a tool for the investigation of multimodal volume data sets. The computation time for the multimodal similarity map of two data sets is approximately twice the computation time of a self similarity map for a data set of the same size. Therefore, a feasible strategy to limit the duration of this pre-processing step is to use downsampled versions of the distance transforms (which are computed at the original data set resolution) for the mutual information computation. Isosurfaces are also good indicators of scalar field behavior, but the volumetric nature of them delivers the obvious inconvenience in case of direct visual analysis.

In a simple case, multi-run data are composed of a set of volumetric time-varying scalar fields. Properly chosen isosurfaces can serve as good descriptors of the entire scalar field carrying information about data features. Hence, we consider them useful for the purposes of the multi-run data analysis. In order to overcome the existing limitations of isosurface comparison, particularly high computational costs, we propose a novel
isosurface similarity measure, which allows for a fast and robust isosurface comparison.

**Analysis of multi-field data**

The visualization of correlations between multiple fields is a big challenge. Sauber et al. [11] present an approach to visualizing correlations in 3D multi-field scalar data. Blaas et al. [12] was one of the first who used sophisticated visualization methods and interaction in the multidimensional attribute space. To explore multidimensional data, Elmqvist et al. [13] presented interactive methods using scatterplots. Woodring and Shen [14] propose using a volume shader where the user is given the ability to easily select and operate on many data volumes to create comparison relationships. Furthermore, they render the contextual information of the volume shader by converting it to a volume tree. The authors visualize the different levels and nodes of the volume tree so that the user can see the results of suboperations. Recently, there has been the approach to couple attribute space clustering with an interactive visual exploration of multivariate volume data. Maciejewski et al. [15] developed multidimensional transfer functions for direct volume rendering using two-dimensional and three-dimensional histograms and density-based clustering within these histograms. Since interactions with the histograms are necessary for visual analysis of the data, their approach is restricted to attribute spaces of, at most, three dimensions. Linsen et al. [16, 17] proposed an approach that can operate on multivariate volume data with higher-dimensional attribute spaces. The attribute space is clustered using a hierarchical density-based approach and linked to a physical-space visualization based on surface extraction. However, the result of the analysis depends on the clustering quality. Recently, the approach was extended by Dobrev et al. [18] to an interactive analysis tool incorporating direct volume rendering. Indeed, Dobrev et al. show that the generated clustering result is often not as desired and propose interactive means to fix the clustering result. Schneider et al. [19] expand their approach for a multi-variate case, but it has similar limitations as the approach for single-variate data [8].
All these approaches are useful to navigate and to group data within one simulation or one data frame, but in case of multi-run simulations we can not estimate the global trends in the scalar fields. However, visual comparison within a set of selected simulations’ frames is still possible, e.g., comparison of final simulations’ states. For that one needs a fast and interactive visualization tool. Most of the existing approaches require considerable time to produce a resulting visualization. We propose to use scatterplots for a visualization of data points’ relations between different data fields and introduce algorithms and tools for “on-the-fly” frequency based progressive rendering of scatterplots for continuous fields with an opportunity to investigate a multi-dimensional attribute space. We also generalize our isosurface similarity measure to allow comparison of entire multi-variate data frames.

### Analysis of time-varying multi-field data

To explore and visualize multivariate time-varying data sets, Lee and Shen [20] presented an algorithm. To identify the temporal trends from a local region, they designed an algorithm to estimate when a trend appears and vanishes in a given time series. Based on the beginning and ending times of the trends, their temporal relationships can be modeled as a state machine representing the trend sequence. However, the speed of the algorithm is quadratic to the number of time steps and hence is the performance bottleneck. For interactive exploration of large multi-time point data sets, the parallel coordinate plots are used successfully. An approach for the data exploration was proposed by Paulovich et al. [21]. With the proposed Part-Linear Multidimensional Projection technique, they project the high-dimensional instances into the visual space in a streaming way, and the streaming projection enables the analysis and visualization of voxels with similar features on massive time-varying data. Akiba and Ma [22] presented an user interface for visualizing time-varying multivariate volume data. This interface is composed of three tightly coupled views characterizing the data in different spaces. Wei et al. [23] operated in a 2D phase space and cluster trajectories to analyze particle data in combustion simulations. In the article by Blaas and Botha [24] techniques for preprocessing using data quantization and compression, and for fast GPU-based rendering of PCPs using joint density distributions for each pair of consecutive variables, resulting in a smooth, continuous visualization are well described. Hao et al. [25] use pixel-based techniques to visualize time-dependent data at multiple resolutions based on importance
values per data interval. The basic idea of their approach is that either data-dependent or application-dependent, display space is allocated in proportion to the degree of interest of data subintervals, thereby guiding the user in perceiving important information, and freeing required display space to visualize all the data. The noted approaches are concentrated on single-run datasets and do not allow to analyze a whole ensemble of simulations and to estimate a system behavior under the influence of input parameters.

For the purposes of multi-run simulation data analysis, a single-run simulation data visualization should allow for easy comparison and convey information about simulation features. We propose to use function plot visualization, because it combines all the necessary features: it is informative, easily comparable and allows for an interactive analysis. Using such visualization one can investigate a single simulation, or add plots to highlight shared data features.

**Analysis of multi-run simulations**

Many tools were developed to work with multi-run data. Potter et al. [26] presented an approach, where a collection of statistical descriptors is used for analyzing ensemble data sets. "Ensemble-Vis" is a framework consisting of a collection of overview and statistical displays linked through a high level of interactivity (Potter et al. [27]). It is focused on a weather forecasting and climate modeling, therefore, it has very specific user interface and input data formats. "Noodles" has been developed to interactively visualize ensemble output and associated uncertainty of a weather event dataset (Sanyal et al. [28]). It has similar characteristics as "Ensemble-Vis". The common disadvantage is an absence of an exploration tool for input data allowing for comparing and grouping the input datasets to pick necessary attributes for the further analysis. Two techniques to support ensemble exploration and comparison were developed by Phadke et al. [29]. These techniques are limited to comparing only a small number of members at any given time, and the pairwise sequential animation technique begins to suffer when more than three members are shown. To estimate an uncertainty of the modeling or input data, Pöthkow et al. [30] proposed a method for quantification of spatial uncertainty of isocontours considering arbitrary spatial correlations of the probability distributions of the input data. The current implementation is limited to multivariate Gaussian distributions and the method is very expensive in terms of computation time. Visualization and
comparative analysis of vector field ensembles were investigated by Jarema et al. [31], but the proposed approach aims only at 2D vector fields. Another approach which aims at visual analysis of vector fields was presented by Ferstl et al. [32]. It allows to plot a streamline variability for characterizing the uncertainty for 2D and 3D data. However, both approaches are not applicable to time-varying simulation data analysis. An interactive approach to enable a continuous analysis of a sampled parameter space with respect to multiple target values was investigated by Berger et al. [33]. It is a suitable approach for a certain frame analysis, but it does not tackle spatio-temporal data. Interactive visual analysis in combination with a complex data model, which supports families of curves in addition to scalar parameters highlighted by Konyha et al. [34]. An advantage of using advanced interaction is that it does not increase the amount of data and the visual complexity, because usually no additional views are necessary. On the other hand, it generates an additional cognitive load, since the user needs to mentally manage the advanced interaction method. In our task we aim to handle general volumetric data with efficient exploration tool for comparative analysis of simulations within a simulations’ ensemble, including visual analysis of impact of simulations’ parameters and initial conditions. Molchanov and Linsen [35] proposed an approach for a visual exploration of patterns in multi-run time-varying multi-field simulation data using projected views. The idea of the approach is to construct an attribute space for all the simulation frames, and project descriptive attribute vectors to a low dimensional space. Then, connect points corresponding to consecutive time steps of one run by line segments, such that each simulation run is represented as a polyline. The main disadvantage of the approach is that there is no unified recipe on how to best define an attribute space for any multi-run simulation, and therefore too much effort has to be put in order to apply it.
We aim to overcome most of the listed general and specific restrictions for a multi-run multi-field simulation data visual analysis, except of current hardware limitations of an average personal computer or laptop. Thus, we propose an approach, which is equally efficient for data of any spatial configuration and independent of application area. The concept of multi-run similarity plots allows for a qualitative and comparative analysis of an ensemble of simulations, combining visualization of an entire data set with information about individual simulation features and parameter space. The proposed similarity measure allows to compare data frames using an entire information about multiple data fields. Also we propose a number of tools to support this technique by coordinated linked visualizations, expanding its applicability to the data analysis on all the data aggregation levels from the whole simulation ensemble to a single data frame.
Chapter 3

Isocontour similarity measure

This chapter is based on the publication:


Isocontour similarity maps are a technique to visualize structural information about volumetric scalar fields based on sampling the field’s range by a number of isovalues and comparing corresponding isosurfaces. The result is displayed in form of a 2D grayscale map that visually conveys structural components of the data field. In this paper, we present a novel way to establish isosurface similarity maps by introducing a quasi-Monte Carlo approach for computing isosurface similarities. We discuss our approach and implementation details in comparison to the state of the art. We show that our method produces significantly lower computational costs, yet it is simpler and more intuitive to use, is more flexible in its applicability, and more robustly generates high-quality results.
3.1 Introduction

Volumetric scalar fields are generated in numerous areas of science, engineering, and medicine. They stem from running simulations, e.g., of some physical phenomena, or taking measurements, e.g., using medical imaging techniques. Isosurface extraction has established itself as one of the key analysis tools for such data fields. Isosurface \( S_c \) of a scalar field \( \phi : \mathbb{R}^n \to \mathbb{R} \) with for an isovalue \( c \in \mathbb{R} \) is a set \( S_c = \{ v \in \mathbb{R}^n | \phi(v) = c \} \).

Looking at isosurfaces one can distinguish different structural components of underlying data, which correspond to different ranges of the scalar values. Manual inspection of all isosurfaces within the field’s range would be a tedious task.

In this Chapter, we build upon a strategy to support interactive volumetric scalar field analysis known as isosurface similarity maps. The concept of isosurface similarity maps was introduced by Bruckner and Möller [9]. They use a discrete set of \( N \) isovalue \( V = \{ v_1, ..., v_n \} \) in a data set to generate a \( N \times N \) matrix \( SM_V(i,j) \) containing an isosurface similarities computed for each combination of isovalue \( v_i \) and \( v_j \). The main idea of the approach is to build a 2D gray-scale map, which visualizes pair-wise distances between all isosurfaces from a sampling of the field’s range, where dark colors correspond to small distances and bright colors to large distances. Looking at such similarity maps, one can easily distinguish ranges of field values that correspond to similar structures. Therefore, these maps help to choose proper volume visualization parameters such as a good selection of isosurfaces or a suitable transfer function.

Our contribution is a new approach for fast and robust extraction of isosurface similarity maps. Our primary goal was to reduce the high computational costs of existing approaches. We replace the concept of using distance transforms as surface descriptors and mutual information estimates for isosurface similarity computations by a quasi-Monte Carlo (qMC) approach for comparing isosurfaces by looking at the enclosed volumes. We compare the two approaches by discussing implementation details, the quality of the produced similarity maps, and computation times. We show that our method is significantly faster, more reliably produces the desired results, and is more intuitive and more flexible to apply.
3.2 Related Work

In addition to their property of being easily visualized and understood, isocontours have proven to be a suitable descriptor for scalar fields [6, 7, 9, 36–39]. Bruckner and Möller in their work [9] present structural information of a volume data set by depicting similarities between individual isosurfaces quantified by an information-theoretic measure. This concept can even be successfully extended to multimodal volume data (Haidacher et al. [10]). One obvious disadvantage of the approach is the considerably high costs of generating the isosurface similarity map. The reported implementation can require several hours of processing time for a single data frame.

Besides the basic purpose of data visualization, the concept has a wide area of potential applications, such as volume quantization and compression, volume segmentation, or multi-dimensional classification. However, for generating similarity maps, some requirements have to be satisfied. First, a proper distance function has to be defined to generate a similarity matrix. Second, since isocontours are defined implicitly, it is important to describe all isocontours with the same accuracy for a fair comparison. Third, the isocontours do not only differ by enclosing areas, but have different shapes, which needs to be taken into account.

3.3 Distance Computation

3.3.1 Mutual Information

Mutual information is a widely used measure of similarity between two random variables. It has been successfully applied to a range of problems [40–42]. A formal definition [43] of the mutual information between two random variables $X$ and $Y$ is given by

$$I(X,Y) = \sum_{x \in X} \sum_{y \in Y} p_{X,Y}(x,y) \log \left( \frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)} \right),$$

where $p_{X,Y}$ denotes the joint probability distribution function of $X$ and $Y$, while $p_X$ and $p_Y$ denote the marginal probability distribution functions of $X$ and $Y$, respectively.
Mutual information can also be expressed in terms of entropy by

\[ I(X, Y) = H(X) + H(Y) - H(X, Y), \]

where \( H(X) \) and \( H(Y) \) denote the marginal entropies and \( H(X, Y) \) the joint entropy of \( X \) and \( Y \). For our purposes, it is more convenient to work with normalized values [44] given by

\[ \hat{I}(X, Y) = \frac{2I(X, Y)}{H(X) + H(Y)}. \]

In order to apply the mutual information measure to isosurfaces, a proper surface descriptor is needed. Bruckner and Möller [9] propose to use for this purpose the distance transform of the isosurfaces [45]. Thus, the distances from any point of the data volume to a pair of the to-be-compared isosurfaces can be considered as random variables \( X \) and \( Y \). The resulting value of the normalized mutual information measure can be interpreted as a similarity measure between the isosurfaces. Similar isosurfaces have a value closer to 1, while dissimilar isosurfaces are expected to have a value closer to 0.

The nature of the distance transform (DT) makes it sensitive to changes in the geometry of the isosurfaces’ shapes. Generally, isosurfaces exhibit an onion peel-like structure, which means that one can expect smooth shape transitions for close-by isovales provided that the isovales correspondent to the same structural component. Conversely, if DTs of a set of isosurfaces are very similar, we can assume that they belong to the same component.

We propose to replace the isosurface similarity computation based on mutual information with an approach that computes isosurface distances by comparing which areas in the data volume are enclosed by them. Hence, we need to integrate characteristic functions of the areas enclosed by isosurfaces. We propose to compute the integrals using a quasi-Monte Carlo approach. The main benefits of such approach are that it is faster than the mutual information extraction, it can be easily used for spaces of any dimension, and it can be used for isosurfaces defined over any spatial data structure ranging from structured grids over unstructured meshes to unstructured point-based data.
3.3.2 Quasi-Monte Carlo Approach

We compute random points that sample the spatial domain uniformly and evaluate the scalar field at these random points. From the interpolated scalar function, we can derive whether the random point lies inside (including boundary) or outside the isosurface. This information is stored in a binary vector, where the dimensionality of the vector is equal to the amount of random points. Hence, the binary vector is a representative descriptor of the isocontour. Note that the descriptor does not only allow for the estimation of the size of the area enclosed by the isocontour, but also captures the location and shape of the enclosed area. Hence, when comparing two isosurfaces with respect to this descriptor, one can estimate how closely the isosurfaces’ location and shape match.

Based on the introduced vectors, we can introduce a distance function between the respective isocontours. Let $A$ and $B$ be isocontours, $M_{A \land B}$ the number of points inside both isocontours (logical and), and $M_{A \lor B}$ the number of points inside of, at least, one of the isocontours (logical or). Then, we define a distance $d(A, B)$ between $A$ and $B$ by the Jaccard distance

$$d(A, B) = 1 - \frac{M_{A \land B}}{M_{A \lor B}}.$$ 

The idea of the distance calculation is illustrated in Figure 3.1. The example shows two isocontours with $M_{A \land B} = 3$ and $M_{A \lor B} = 15$, i.e., distance $d(A, B) = 0.8$. The accuracy or complexity of the calculations can be adjusted by increasing or decreasing the amount of random points. Due to the randomness, the error of the approach depends only on the amount of points and not on the shapes of the isocontours. Obviously, the algorithm can be applied to scalar fields of any dimension and of any spatial data structure.

![Figure 3.1: Example of two isocontours with distance 0.8.](image-url)
3.3.3 Sensitivity of the Measures

Due to different nature of the isosurface descriptors (DT vs. binary vector encoding inside-outside property of random points), the resulting similarity maps may differ substantially. To illustrate the features of the measures, we generated five synthetic 2D data sets (shown as 2D images) and corresponding similarity maps using both methods, see Figure 3.2.

The first example is shown in column (a). The data field contains three objects represented by different isovalue ranges. The left-bottom corner of the 2D field contains a white circle, while the right-top corner contains dark-gray cross placed inside a light-gray ring. Looking at the similarity map (3) generated by our qMC approach, we can easily distinguish three detached dark squares on the diagonal of the map, which correspond to the three different objects. However, in the similarity map (2) generated by the mutual-information approach, one can only distinguish two squares. The reason for that is that the DTs for lower isovalue look very similar due to the close-by positions of the ring and cross.

The second example in column (b) of Figure 3.2 shows the same data set, only that the cross has moved to a different location outside the ring. Now, we can observe that the mutual-information approach (2) recognizes all three objects, as the corresponding DTs now differ substantially due to big distances between the objects. The qMC approach (3), on the other hand, remains exactly the same as in column (a). Hence, we can conclude that our qMC approach is less sensitive to the spatial distribution of (non-overlapping) features and rather captures their shape differences.

The third example shown in column (c) of Figure 3.2 represents three concentric circles with increasing intensity, while the fourth example in column (d) represents the same data set after adding noise. The resulting similarity map obtained by mutual information (2) for the noise-free data set (c) allows us to weakly recognize three square-like structures along the main diagonal, but there is an additional wave-like artifact covering the whole map. This artifact is caused by repeating discrete circle patterns during smooth isovalue transitions. For the noisy data (d), the similarity map (2) even exhibits another artifact in form of artificial horizontal and vertical stripes. Our qMC
Figure 3.2: Examples of differences between the distance measures. In row (1) five different synthetic data fields are shown. Row (2) depicts the corresponding similarity maps generated using mutual information measure. Row (3) represents the corresponding similarity maps generated using quasi-Monte Carlo approach.

measure (3), instead, produces in both cases the expected and desired result, i.e., exposing the three different objects. Hence, the second conclusion is that our approach is less sensitive to small changes and, therefore, is more robust against noise and extraneous artifacts.

Finally, in column (e) of Figure 3.2, the data set also contains three concentric circles, but intensity values are not increasing anymore plus the central circle is very small. For our qMC approach (3), the difference between the big white ring with and without the small circle is negligible, while in terms of the DT this difference totally changes the picture. This confirms, again, the second conclusion from above.
3.4 Implementation

When using the mutual-information measure for isosurface similarity, there are many implementation details that can affect the similarity map calculation, in particular, the computation time. Hence, it is worth discussing some of the implementation parameters.

The first important parameter is the resolution of the DT. Due to high computation costs, Bruckner and Möller [9] proposed to downsample the resolution, as even at reduced resolutions the DT captures well the characteristics of an isosurface. Since they provided computation times for all examples using a resolution of about $64^3$, we decided to do the same for a fair comparison. Note that downsampling the DT is different from downsampling the data field, as a downsampled version of the original DT is used.

After DT calculation, it is possible to compute the mutual information or distances between the isosurfaces. In order to calculate marginal and joint entropy, we have to build a joint histogram. When doing so, we have to define a resolution of the histogram and its range. Like Bruckner and Möller we use a fixed histogram resolution of $128^2$ throughout this paper. There are many ways to define the range, such as going from 0 to the maximum possible distance (e.g., main diagonal of data volume) or defining a certain range for each pair of isosurfaces. Depending on the data, we observed that different choices can lead to varying quality of the resulting similarity maps. We calculate the minimum and maximum distances among all DTs and use the resulting range for all pairs.

Unlike Bruckner and Möller we did not use a CUDA-based implementation of the joint and marginal entropies by Shams and Barnes [46]. The reasoning is that with a fixed number of used isovalues (256), a fixed joint histogram resolution ($128^2$), and using a downsampled DT resolution, the CPU implementation was actually faster than the GPU implementation in the comparisons we conducted. This is due to additional costs for memory transfer and due to the GPU initialization. The calculated DTs need to be stored in system memory or, in case of very high resolutions, on hard disk.

In contrast to the mutual information approach by Bruckner and Möller, our qMC approach has only one parameter, which is the number of random points. Intuitively, the more random points we use, the higher accuracy we achieve. In order to achieve a
high performance, we propose to pass all the calculations to a GPU. There are two main steps of our similarity map computation: calculating the descriptive binary vectors for all the isosurfaces and comparing them to fill the map with distance values. Both steps are well parallelizable and can be easily implemented with high efficiency.

The amount of required memory and time for performing the qMC approach does not depend on the data resolution (obviously, except for data loading itself). For example, there is the same amount of memory required for similarity map extraction from data of resolution $32^3$ or of resolution $512^3$, if the same number of random points is used. Another advantage is that no explicit isosurface extraction is required, as we only need to interpolate the values of the data field to make the inside-outside decision for each random point. In this paper, we used 32,768 random points which ensure enough accuracy for all the examples. All methods were implemented in C++, while using CUDA for GPU-based functions.

3.5 Results and Discussion

3.5.1 Similarity Map Results

To test our approach on real data, we chose three datasets with different characteristics and from different application domains. Figure 3.3 shows the results for the Crossed Rods dataset. In column (a), two similarity maps corresponding to qMC (1) and mutual information (2) measures are shown. Our similarity map (1) exhibits more structures than (2). For a detailed investigation we selected six isovalues corresponding to the visible structures (marked by colored crosses) and rendered corresponding isosurfaces using the Marching Cubes approach (Lorensen and Cline [47]), see columns (b-d). We observe that each of them differs from the previous one by losing a part of the construction. Hence, our similarity map (1) captured the structure somewhat better.

For a medical imaging data analysis, we used a CT scan of a man’s head, see Figure 3.4. The structures captured by mutual information technique (b) are also visible in our similarity map (a). Moreover, the right-top square area in (b) actually contains some more complicated structure, which is only captured in our similarity map (a). To
3.5. RESULTS AND DISCUSSION

Figure 3.3: Results for Crossed Rods dataset: In column (a) similarity maps are shown, where (1) is the result when using qMC and (2) when using mutual information measures. Columns (b-d) represent isosurfaces corresponding to the selections in the qMC similarity map.

Figure 3.4: Results for CT head scan: (a) and (b) shows similarity maps generated using qMC and mutual information, respectively. Isosurfaces (c, d) correspond to the selection in the similarity map (a).

validate our result, we picked respective isovalues and rendered the corresponding isosurfaces (c, d). Due to relatively small size of the jaws when compared to the whole head, the isosurfaces are judged to be similar when using a DT, but they are judged to be different when considering enclosed volumes.

Since our method is applicable to data of any spatial configuration, we tested it on astrophysical SPH simulation data of a two-stars system, see Figure 3.5. The stars have different masses and sizes. When investigating the internal energy field, we can
3. ISOCONTOUR SIMILARITY MEASURE

Figure 3.5: Results for unstructured SPH dataset (White Dwarfs): (a) similarity map generated using qMC. (b-d) Isosurfaces corresponding to selection in (a).

recognize them by occupied field ranges. The similarity map exhibits three regions (a). Figures (b-d) show the corresponding isosurfaces using splat-based rendering (Linsen et al. [48]), which represent the expected result. The considered gridded data sets are from The Volume Library, the SPH data were provided by Marius Dan and Stephan Rosswog.

3.5.2 Computation Times

Our approach is significantly faster than the mutual information approach by Bruckner and Möller [9]. For the presented examples, computation times are listed in Table 3.1. We executed the algorithms on a laptop with Intel Core i7-3630QM, NVIDIA GTX 660M, and 8 GB DDR3.

Table 3.1: Comparing computation times for the mutual information approach to our qMC approach.

<table>
<thead>
<tr>
<th>Data</th>
<th>Original resolution</th>
<th>DT resolution</th>
<th>Random points</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>qMC (GPU)</td>
</tr>
<tr>
<td>Figure 3</td>
<td>64x64x64</td>
<td>64x64x64</td>
<td>32,768</td>
<td>0.41 s</td>
</tr>
<tr>
<td>Figure 4</td>
<td>128x256x256</td>
<td>32x64x64</td>
<td>32,768</td>
<td>0.48 s</td>
</tr>
<tr>
<td>Figure 5</td>
<td>39,718 particles</td>
<td></td>
<td>32,768</td>
<td>0.64 s</td>
</tr>
</tbody>
</table>

Using our approach, we achieved 430-1,330 times faster similarity map computation for downsampled DT resolutions using GPU, and 50-130 times using only CPU. With increasing resolution our approach keeps the performance, as it only depends on the number of random points, while the mutual information approach has super-linearly
increasing computation times leading to even higher differences between the two approaches. Note that the small difference between qMC times for the first and second example is caused by the different numbers of succeeding conditional statements and not by the different resolutions. In case of SPH data, the more complicated field interpolation leads to higher computation times.

3.6 Conclusion

We presented a novel approach for a similarity map computation based on a quasi-Monte Carlo method. We compared our approach for calculating distances between isosurfaces with the state of the art and discussed important features as well as implementation details. For all examples, our similarity maps presented the structures in the data fields better or, at least, equally good, while being hundreds of times faster and using less system resources. Due to insignificant hardware requirements, our approach is also suitable for most devices, even those with a single core architecture. Hence, we could overcome the main drawbacks of similarity map computation based on mutual information.
Chapter 4

Progressive rendering of continuous scatterplots

This chapter is based on the publication:


Continuous scatterplots are a consistent tool for the visual representation and exploration of continuous multivariate data defined on a continuous domain. Due to the complexity of the construction algorithm, application of continuous scatterplots is limited in terms of data size and screen resolution when interactive frame rates are desired. Progressive rendering is a paradigm of displaying an approximative visual outcome early on, which iteratively and incrementally gets improved until convergence to the final result is reached. This approach maintains the interactivity of the system and allows the user to make decisions immediately, i.e., much earlier than the end of the computation process. We propose a method for progressive rendering of continuous scatterplots based on a Fourier representation. By iteratively advancing from low to high frequencies and inverting the spectrum representation after each iteration, a series of scatterplots converging to the final result is generated and rendered. We demonstrate that this convergence is monotonic and that the proposed approach is more efficient than
state-of-the-art methods, i.e., we can faster produce high-quality approximations. We propose to embed this idea into a hybrid approach which allows balancing the trade-off between quality of the image appearing first and its computation time. The proposed algorithms were implemented on the GPU.

The theoretical base of the approach for progressive rendering of continuous scatterplots was developed and introduced by Vladimir Molchanov. Development and implementation of the computational algorithms using GPU for the rendering were done by Alexey Fofonov. This constitutes a contribution in frames of the thesis. Successful combination of the methodological base and efficient usage of the computational resources allows to speed-up the approach for two orders of magnitude depending on the resolution of the rendering output comparing to the state of the art technique.

4.1 Introduction

Most volumetric scalar fields, no matter whether they have been measured or simulated, are assumed to be continuous or, at least, piecewise continuous. Examples include fields of temperature, density, pressure, salinity, etc. Seldom, these scalar fields can be described analytically. In most cases, data values are acquired by sampling the scalar field at discrete locations in space. In order to reproduce their continuous nature, an interpolation method is involved.

Statistical properties of scalar fields can be explored via histograms. Pair wise relations of two such properties are usually represented in form of scatterplots. Multiple scatterplots can be investigated in form of a scatterplot matrix. All these techniques are well known and have been used for decades by operating on the values at the discrete sample locations. However, such an important property as continuity is being lost at this step.

Recently, new methods appeared which address the problem above by creating continuous representations of attribute spaces. They propose continuous histograms [6, 7], continuous parallel coordinates [49, 50], and continuous scatterplots [51, 52]. The idea behind these approaches is to perform a mapping of parts of the volume rather than just
discrete samples when projecting to the visual domain. The distribution of attributes in the volume parts is reconstructed by means of interpolation.

 Obviously, much more computational efforts are required to produce a continuous scatterplot when compared to its discrete analog. Therefore, there is a clear need for efficient algorithms showing high performance when applied to large datasets and high-resolution outputs. One approach to design an interactive tool is to exploit the idea of progressive rendering. This idea is based on an iterative generation of visualizations with increasing quality which converge to the final result. The gain is to provide the user with preliminary but still informative outcomes at interactive rates. Based on the preliminary result, the user can decide whether it is worth to wait for the final result or whether the interactive query shall be modified.

 The essential properties of progressive rendering algorithm which we target in our research can be summarized as follows:

 1. The key features of the final result shall appear first.

 2. The change of intermediate results when stepping from iteration to iteration shall vanish. In other words, the new contributions to the final image decrease in each iteration.

 3. The result obtained at the previous step shall be effectively used when performing the next iteration. In other words, computations already made shall not be repeated.

 Our work was inspired by the continuous scatterplots technique proposed by Heinrich et al. [50]. In this algorithm, the physical volumetric domain is represented as a collection of Gaussian kernels centered at data samples. When mapping to the attribute domain (i.e., the domain spanned by the two dimensions of the scatterplot), each kernel appears as an elliptical splat called a footprint. The geometry of a footprint is entirely defined by the parameters of a locally linearized mapping, which allows for the pre-computation of footprints in form of splat textures. To handle large datasets, Heinrich et al. propose to split the data into chunks, produce an individual plot for each portion of the data, and then iteratively combine the plots over several frames using alpha
blending. The result is a progressively updated image which is based on the amount of data already processed. So, the work by Heinrich et al. uses the general idea of a progressive rendering. However, using their approach, it is not easy to fulfill the first two requirements formulated above. The result may be highly affected by the way, how the data are split into chunks. Sophisticated methods may help to make a wise decision, but they may also break the interactivity of the application.

Our main contribution is a novel approach for progressive rendering of continuous scatterplots based on their frequency representation (spectrum). This kind of representation is obtained by applying a Discrete Fourier Transform (DFT) to the scatterplot’s density distribution. The key idea is as follows: Small-sized footprints have bad (slowly decaying) spectra but can be efficiently plotted directly into the attribute space density plot (continuous scatterplot). Large-sized footprints slow down the computations when blended in the attribute space, but they have nice frequency representations. Hence, we propose to split the physical space kernels into two classes: Those having small footprints (few pixels support) and those whose footprints have good Fourier representations. The former are plotted directly while the latter are processed iteratively, progressing from the lowest most meaningful modes to the highest frequencies with vanishing contributions.

Besides that, we extend the existing method to unstructured volumetric data and explore the performance and error dependence on most key parameters. We implemented the method by Heinrich et al. [50] and our own algorithm entirely on the GPU (using CUDA) for better evaluation of their applicability and efficiency.

4.2 Related Work

The idea of progressive image generation has been known for decades [53]. Progressive methods find their application in many visualization and computer graphics problems, e.g., ray tracing [54], global illumination [55], or volume rendering [56]. One of the most prominent examples is surface renderings where progressive meshes [57] can be used in case of irregular meshes and subdivision surfaces in case of regular patches or semi-regular meshes. Also, isosurface extraction from progressively refined tetrahedral
or pyramidal meshes (e.g., [58]) and isosurface smoothing [59] are common examples in volume visualization.

Spectral analysis is used to improve sampling of the rendering integral in volume rendering [60]. A wide range of progressive spectral methods [61], especially based on the Fast Fourier Transform (FFT), are very common in image processing, e.g., for image registration [62] or for characterization of brain fibers’ shape [63].

Scatterplots are a well-known tool for multidimensional data visualization and analysis [13]. Recently, an idea of continuous scatterplots was developed in a series of works by Bachthaler, Heinrich, and Weiskopf. Initially applied for mapping tetrahedra by using a linear interpolation of attributes within them [51], the approach was generalized to regular rectangular grids with an arbitrary interpolation method [52], where recursive subdivision of cells was exploited. Later, isotropic density functions were used to decompose the whole volume into a system of overlapping spheres [50]. Splatting is performed for the generation of progressively sampled intermediate images which are then combined to produce the final continuous scatterplot. The progressive rendering component of the algorithm is based on partitioning the volumetric data into small chunks which are fast to operate. Intermediate images are produced for down-sampled or even freely re-sampled data. For instance, it is suggested to find new samples positions from low-discrepancy sequences [64].

The idea of image space footprint computation of volumes has been developed by Westover [65]. Feng et al. [66] use Gaussian footprints to visualize data samples uncertainty. Lehmann and Theisel [67] developed a method to find and highlight discontinuities in continuous scatterplots.
4.3 Background

Before we describe our approach in detail, we would like to provide the respective background. For the construction of the continuous projections, we look into the settings of the involved spaces (a volumetric physical domain and a 2D attribute domain which is visualized). We discuss some technical aspects of the method by Heinrich et al. [50] and, in particular, provide useful generalization by allowing the lengths of spatial kernels to vary from sample to sample.

4.3.1 Basic Terms and Notations

Let \( \tau(x) \) be a multivariate multidimensional function with \( m \) variables defined over \( n \) dimensions, i.e., \( \tau : \mathbb{R}^n \rightarrow \mathbb{R}^m \). It is sampled at positions \( \{x_i\} \subset X \subset \mathbb{R}^n \) mapping them to the set \( \{q_i\} \subset Q \subset \mathbb{R}^m \), \( i = 1, \ldots, N \). Figure 4.1 illustrates our notations. There exists a scalar non-negative function \( s_{\text{overall}}(x) \) defined on \( X \) which is called the spatial density function. It represents the importance of data in the volume and is usually set to be constant. For later consideration it is useful to approximate the density function by a weighted sum

\[
s_{\text{overall}}(x) = \sum_{i=1}^{N} w_i s_i(x),
\]

where every individual kernel \( s_i(x) \) is obtained from a compactly supported shape function \( s(x) \) by

\[
s_i(x) = s \left( \frac{x - x_i}{h_i} \right),
\]

where \( h_i \) defines a scaling radius of \( s_i(x) \). In our work we focus on isotropic kernels such that \( s(x) = s'(||x||) \). Our goal is to find proper weights \( w_i \) to achieve \( s_{\text{overall}}(x) \approx \text{const.} \). Since this condition simply means that all parts of domain \( X \) are equally important, the exact value of the constant does not matter. Then, if one integrates the relation (4.1) over the volume, one deduces

\[
C_1 \cdot \text{Volume}(X) = \sum_{i=1}^{N} w_i \int_{\mathbb{R}^n} s_i(x) \, dx = C_2 \sum_{i=1}^{N} w_i h_i^n.
\]
4. PROGRESSIVE RENDERING OF CONTINUOUS SCATTERPLOTS

Figure 4.1: Physical volume $X$ is mapped by $\tau$ to the attribute domain $Q$. A spatial density $s_i$ is defined in a spherical neighborhood of sample $x_i$, which has an elliptical footprint in $Q$. Scalar function $\sigma_i$ denotes density on the footprint.

with constants $C1$ and $C2$. This means that $w_i h_i^n$ stand for volume fractions associated with samples. Therefore, one takes $w_i = 1$ for all $i$ to get $s_{\text{overall}}(x) \approx \text{const.}$

In the following, we study the case $n = 3$ and $m = 2$ as the most practically important one, since it is common practice to investigate pairs of attributes defined over a volume by means of scatterplots.

Let $X_i$ be the support of the density function $s_i(x)$. The construction of a continuous scatterplot is based on additive mapping of all $X_i$ weighted by $w_i s_i(x)$ by means of $\tau(x)$ to the attribute domain $Q$. Recall that discrete scatterplots map the data sample by sample resulting in the set $\{q_i\}$. A continuous scatterplot is composed by a collection of images of $X_i$. It is shown in [50] that spherical kernels $s_i(x)$ result in elliptical footprints. Each footprint is equipped with a density distribution $\sigma_i(q)$, which represents a counterpart of the spatial density $s_i(x)$, see Figure 4.1. Hence, a continuous scatterplot $\sigma_{\text{overall}}(q)$ is the direct sum of densities $\sigma_i(q)$.

The key relation between the two density functions is given by

$$\int_{X_0} s_{\text{overall}}(x) d^n x = \int_{Q_0} \sigma_{\text{overall}}(q) d^m q, \quad (4.2)$$

for any $X_0 \subset X$ and $Q_0 = \tau(X_0)$. This condition uniquely defines $\sigma_{\text{overall}}(q)$. Equation (4.2) with $X_0 \equiv X$ and $Q_0 \equiv Q$ is called total mass conservation.
4.3.2 Direct Approach for Generation of Continuous Scatterplots

Following the method proposed in [50], a spherical neighborhood of sample \( x_i \) of radius \( kh_i \) is mapped as an elliptical splat centered at \( q_i = (q_1(x_i), q_2(x_i)) \) using a linear approximation of the mapping \( \tau \) in a neighborhood of sample \( x_i \). The density \( \sigma_i(q) \) is non-zero within this footprint. Besides its center, a footprint is uniquely defined by its extents \( S_{i,x} \) and \( S_{i,y} \) (semi-axes of the screen-space ellipse) and its rotation angle \( \theta_i \). These parameters are defined as follows (cf. [50, 65]):

\[
S_{i,x} = kh_i \sqrt{\lambda_1}, \quad S_{i,y} = kh_i \sqrt{\lambda_2},
\]

\[
\cos \theta_i = \frac{b}{\sqrt{b^2 + (\lambda_1 - a)^2}},
\]

where \( k \) is a global smoothing parameter discussed below, \( h_i \) is a local support size which may vary from sample to sample and reflect the samples’ density in the neighborhood of the \( i \)-th instance,

\[
\lambda_1 = \frac{a + c + e}{2}, \quad \lambda_2 = \frac{a + c - e}{2}, \quad e = \sqrt{(a - c)^2 + 4b^2}
\]

and

\[
a = \nabla q_1(x_i) \cdot \nabla q_1(x_i), \quad b = \nabla q_1(x_i) \cdot \nabla q_2(x_i),
\]

\[
c = \nabla q_2(x_i) \cdot \nabla q_2(x_i),
\]

where \( \cdot \) stands for the scalar product. Note that the computation of \( \cos \theta_i \) can be numerically instable for small \( b \) when applying Equation (4.4) naively. If \( |b| < \varepsilon \) for some threshold \( \varepsilon > 0 \), the splat is not rotated, i.e., \( \theta_i = 0 \). Thus, one can explicitly set \( \cos \theta_i = 1 \).

The user is allowed to change the value of parameter \( k \) which simultaneously scales the extents \( S_{i,x} \) and \( S_{i,y} \) of all footprints. Thus, the global smoothness of the continuous scatterplot can be controlled: For small values of \( k \) the result is almost a discrete plot, while large values of \( k \) make the resulting image blurred. This scaling is equivalent to the respective change of support sizes of spatial kernels \( s_i(x) \). Here, their supports shrink to samples \( x_i \) as \( k \) vanishes and unboundedly grow when \( k \to \infty \). Note that for
larger values of \( k \), the number of locally overlapping spatial kernels increases.

All elliptical footprints are added as textures to the final image. In the following we refer to this procedure as a *direct (splatting) approach* in contrast to the progressive method proposed in this paper. After a normalization step which makes the range of the accumulated density \( \sigma_{\text{overall}}(q) \) equal to \([0, 1]\), a transfer function can be applied. The main bottleneck of the method is related to the rendering of large splats (relative to the screen resolution), since their contributions to many pixels have to be computed. This situation occurs in particular when producing high-resolution images, when zooming into a smaller region of the plot, or when choosing high values of the global smoothing parameter \( k \).

### 4.3.3 Progressive Approach

To overcome the issue of insufficient efficiency of the method for interactive visual analyses, a progressive splatting approach was proposed in [50]. When splitting the input data into small portions and operating on them separately, the number of operations per chunk is reduced. It makes it possible to generate images for each chunk faster.

Gradual accumulation of the generated images results in a progressively changing continuous scatterplot. However, every intermediate result intrinsically depends on which part of the data is already processed and which not. In other words, it depends on the order in which the chunks are accumulated. To illustrate this effect we generated a "Tornado" dataset, courtesy of [68], sampled at \( 40^3 \) regularly distributed nodes. Velocity magnitude and the \( z \)-component of the velocity field are taken as dimensions of the scatterplot. Intermediate results after operating 25\%, 40\% and 50\% of samples as well as the final result are shown in Figure 4.2. All four images are qualitatively very different, so that there is no monotonic behavior of the results along iterations. Hence, this unpredictability of the intermediate results will make the user wait until all computations are done, which destroys the essence of progressive rendering (see Requirements 1 and 2 formulated in the Introduction). Of course, the intermediate images depend on how the chunks of data have been generated and a different choice may have produced qualitatively better approximations early on, but the issue of not knowing whether the
Figure 4.2: Dependence of intermediate images in progressive splatting on the fraction of processed data (“Tornado” dataset with $40^3$ gridded nodes). Even when half of the dataset is presented in the continuous scatterplot, the user cannot fully rely on the shown image, since the final result may differ significantly.

intermediate results reflect the final result well is apparent.

We also note that the normalization of the density, which is necessary before a transfer function can be applied, is very sensitive to the skipping of data portions. In fact, skipping a few samples contributing to the pixel with the highest density obviously changes the normalization factor, which affects the appearance of the whole picture, although the transfer function remains the same.

The situation becomes even worse when dealing with unstructured spatial datasets, which are discussed in the next section. The reason is that there is no standard way
of ordering data in memory and splitting data into chunks can be absolutely arbitrary. Note that even if the data are regularly sampled, after a zooming operation, the data to be displayed generally do not belong to any spatially regular structure. These considerations motivated us to design an approach, where all samples are handled uniformly and simultaneously and where no particular spatial arrangement of the samples is assumed.

Our solution to the issue above relies on the spectral representation of continuous scatterplots. Instead of splitting the data, the progressive part of our approach is based on a consecutive computation of the Fourier frequencies advancing from the lowest most contributing modes to the highest less important ones. In the following section we show that large splats have a nice (rapidly decaying) representation in the frequency domain that makes it possible to significantly speed up the construction of continuous scatterplots.

We note that spatially unorganized data can be stored in a file in arbitrary order. Thus, when applying the progressive splatting idea from [50], the distribution of samples among chunks of data is very accidental. This fact leads to the effect shown in Figure 4.3. Shown is a dataset representing an astrophysical simulation of a binary system consisting of two White Dwarfs. The dimensions of scatterplot are the internal energy values and temperature at samples’ positions. Images for 33.3% and 50% of the samples provide a poor approximation to the final result shown in Figure 4.6 (e).
4.3.4 Spatially Unstructured Data

We generalize the approach of Heinrich et al. [50] by allowing the local scales to vary from sample to sample. It makes it possible to apply the direct approach to irregularly sampled data. In order to define an individual scale \( h_i \), we estimate the local density of samples: In the regions, where samples are dense, \( h_i \) is small, while sparsely distributed \( x_i \) yield large \( h_i \). Only the relative variation of scales is important, since the absolute magnitude can be changed by adapting parameter \( k \). For regular data, the samples’ density is constant everywhere, hence, all \( h_i \) are equal.

The ability to handle irregular data is very valuable for practical applications. For instance, various scientific measurement techniques and particle-based numerical methods may produce large multivariate unstructured spatial datasets which are to be visualized and analyzed. Of course, it is always possible to re-sample unstructured data regularly, but this comes at the cost of introducing interpolation errors and losing spatial adaptivity which may affect the result. Therefore, it is advantageous to operate on the original data.

4.4 Spectral Representation of Continuous Scatterplots

For our purposes, it is important that density \( \sigma_i(q) \) in the footprint domain can be computed from some template kernel function \( \sigma(q) \) via an affine transformation composed by scaling, rotation, and translation, i.e.,

\[
\sigma_i(q) = \alpha_i \, \sigma(L_i(q - q_i)),
\]

\[
L_i = \begin{pmatrix} S_{i,x}^{-1} & 0 \\ 0 & S_{i,y}^{-1} \end{pmatrix} \cdot \begin{pmatrix} \cos \theta_i & \sin \theta_i \\ -\sin \theta_i & \cos \theta_i \end{pmatrix}.
\]

In other words, to compute \( \sigma_i(q) \) at some location, one has to perform translation by \( q_i \), rotation by angle \( \theta_i \), scaling by \((S_{i,x}, S_{i,y})\), and then evaluate the shape-function \( \sigma \) at the obtained location. This is just an analytical expression of the splatting, where a pre-computed texture containing values of \( \sigma(q) \) is placed at the right position in the frame with right scaling and right orientation resulting in the footprint \( \sigma_i(q) \). The coefficient
\( \alpha_i = w_i h_i^3 \cdot S_{i,x}^{-1} \cdot S_{i,y}^{-1} \) serves to conserve the mass associated with the \( i \)-th sample:

\[
\int_{Q} \sigma_i(q) \, d^2 q = \alpha_i \int_{Q} \sigma(L_i(q - q_i)) \, d^2 q
\]

\[
= \frac{\alpha_i}{|L_i|} \int_{Q} \sigma(p) \, d^2 p = w_i h_i^3 \int_{\mathcal{X}} s(y) \, d^3 y = w_i \int_{\mathcal{X}} s_i(x) \, d^3 x.
\]

Here we assumed that the \( L_1 \)-norms of template functions \( s \) and \( \sigma \) are equal and used

\[
|L_i| = S_{i,x}^{-1} \cdot S_{i,y}^{-1}.
\]

Summation over \( i \) provides the total conservation of mass (4.2).

The final density distribution is given as a composition of individual contributions of all samples, i.e.,

\[
I(q_1, q_2) = \sum_{i=1}^{N} \sigma_i(q_1, q_2).
\]

Here and further we use \( I \equiv \sigma_{\text{overall}} \) for short. Now we examine the density function in the frequency domain. By the linearity property, the Fourier transform of \( I(q_1, q_2) \) reads

\[
\mathcal{F}[I](u, v) = \sum_{i=1}^{N} \mathcal{F}[\sigma_i](u, v).
\]

It is known that scaling, rotation, and translation operators become scaling, rotation and modulation after the Fourier transform, correspondingly. In particular,

\[
\mathcal{F}[f(Ax)](q) = |A^{-1}| \cdot \mathcal{F}[f(x)](A^{-1} q).
\]

Thus,

\[
\mathcal{F}[\sigma_i](u, v) = e^{i(uq_1(x_i) + vq_2(x_i))} \alpha_i |L_i^{-T}| \cdot \mathcal{F}[\sigma](L_i^{-T}(u, v)),
\]

with

\[
L_i^{-T} = \begin{pmatrix}
S_{i,x} & 0 \\
0 & S_{i,y}
\end{pmatrix} \cdot \begin{pmatrix}
\cos \theta_i & \sin \theta_i \\
-\sin \theta_i & \cos \theta_i
\end{pmatrix}.
\]

Finally,

\[
\mathcal{F}[I](u, v) = \sum_{i=1}^{N} w_i h_i^3 \cdot e^{i(uq_1(x_i) + vq_2(x_i))} \cdot \mathcal{F}[\sigma](L_i^{-T}(u, v)).
\]

(4.5)

When the spectral representation \( \mathcal{F}[I] \) is computed, one can apply the inverse Fourier
transform to obtain the continuous scatterplot. Practically, we use the discrete Fourier transform rather than the Fourier transform itself. This approximation introduces errors which vanish if the resolution of textures (number of frequencies taken into account) grows. We also need to comment on the periodicity of the discrete Fourier transform: Footprints located close to the boundary of the computed region may have their parts appearing at the opposite part of the boundary instead of being cut. To overcome this effect, the region should be extended to reserve additional space for such footprints.

4.5 Progressive Algorithm

Our complete computation of spectral representation $\mathcal{F}[I]$ is much slower than the standard splatting-based technique for continuous scatterplot generation. Obviously, every footprint has usually non-zero contributions to all frequencies $(u, v)$, whereas the splat may have very small size. Thus, if the resolution of the screen is $R_u \times R_v$, complexity of $\mathcal{F}[I]$ computation is $N \times R_u \times R_v$. Recall, that this representation has to be transformed by the inverse FFT and not the full resolution can be shown, since some border region of the texture is reserved to eliminate the periodicity effect of DFT. However, there is a strong side of $\mathcal{F}[I]$.

It is well-known that the decay of Fourier coefficients depends on the smoothness of a function: The smoother the kernel $\sigma(q)$, the faster the decay of $|\mathcal{F}[\sigma](u, v)|$. In fact, for many footprints, high frequencies does not significantly affect the result and therefore may be neglected. Based on this observation, we propose the following progressive algorithm for continuous scatterplot construction: Evaluate $\mathcal{F}[I](u, v)$ for some low frequencies, perform the inverse FFT and render the result as the first approximation to the final picture. In subsequent iterations higher frequencies are added to the Fourier representation of the scatterplot and the rendered picture is updated after performing the inverse FFT. Since high frequencies of $\mathcal{F}[I]$ computed at later iterations are composed by vanishing contributions from individual splats, later iterations have generally less impact on the final result for smooth kernels. Thus, Requirements 1-3 in the Introduction are fulfilled. The number of frequencies computed in each iteration depends on the hardware performance and the data size. It can be fixed or vary from one step to another.
Figure 4.4: (a) Overcoming the effect related to the periodicity of FFT. Boundary part of computed texture (grey) is cut and will be not rendered to eliminate periodical extensions of footprints. (b) Fourier representation is computed progressively. At the first iteration only main (lowest) frequencies are considered (red square). At subsequent iterations higher modes are added. Inverse FFT is applied and a rendering step is performed after each iteration.

Figure 4.4 (b) illustrates the progressive part of the algorithm.

4.6 Hybrid Algorithm

Large footprints have fast decaying Fourier spectra and therefore can be very efficiently and with high precision represented by a few lowest frequencies. Small footprints can be fast rendered directly to the scatterplot. Here "small" means that at least one of the extents $S_{i,x}$ and $S_{i,y}$ is less than a prescribed threshold. To profit from the best of both worlds, we divide all footprints into two groups according to their size and operate accordingly. The grouping is based only on values $S_{i,x}$ and $S_{i,y}$, i.e., no additional computations are needed. Moreover, the critical size of a splat below which it is labeled as small, depends only on the screen resolution. Since parameter $k$ scales all splats, its value affects the grouping.

The idea of the hybrid approach is illustrated in Figure 4.5. We used the “Bucky Ball” dataset (courtesy AVS, USA) with $32^3$ gridded samples. Small and large splats rendered by means of the direct method and their spectral representation, respectively,
4.7 Implementation

In this section we provide some details on the GPU implementation of the proposed algorithms. We assume that a spatial kernel template (e.g., Gaussian but can be any non-negative isotropic function) and its Fourier counterpart with sufficient resolution are pre-computed and stored as 1D floating-point textures. Note that the Fourier transform of the kernel is real-valued.

Next, we create empty textures reserved for the continuous scatterplot, its spectrum, and an auxiliary texture. For efficiency of the FFT, it is desired to take their sizes $R_u$ and $R_v$ as powers of 2 (all textures have to have same resolution). The spectral texture has to be complex-valued, the two others may be real-valued. Then we choose the parameters of the view-port in order to cut the reserved border of the scatterplot texture, which is shown in grey in Figure 4.4 (a). Note that all centers of footprints should belong to the visualized lower-left white rectangle, if no zooming is performed.

During the first iteration of the method, the geometric parameters of footprints ($S_{i,x}$, $S_{i,y}$, $\cos \theta_i$) are computed. Simultaneously, splats are sorted according to their size.

Figure 4.5: Hybrid approach for continuous scatterplot computation (Bucky Ball dataset): Small splats are rendered directly (a), large splats are computed via their spectral representation (b), and both textures are added to produce the final result (c). Result is compared to direct rendering of all splats (d). Color scheme is shown (e).

see Figure 4.5 (a) and (b). Both plots are combined to produce the final result (c) which is very close to the scatterplot computed by direct splatting (d).
Small ones are sent for direct splatting to the auxiliary texture. This texture will remain unchanged from this point on. If we want to reproduce the direct splatting approach, we just label all splats as small.

Then, the progressive part of the algorithm starts. The lowest frequencies of the Fourier texture are computed accumulating contributions from the large footprints according to Formula (4.5). Here, the number of frequencies found during one iteration depends on the hardware architecture. It is desired but not obligatory to completely fill rectangular parts shown in red, magenta and blue in Figure 4.4 (b) at every next iteration. The symmetry of filling may reduce the noise appearing at intermediate steps.

After each iteration, we map the Fourier texture to the scatterplot texture by means of the inverse FFT. Then, the auxiliary texture with directly collected small splats is added to the scatterplot texture. A normalization step follows. It scales the density values to the range \([0, 1]\). Now a transfer function can be applied and the current result can be shown.

Starting from the second iteration, the filling of the Fourier template continues from the point it stopped at the previous step. The inverse FFT, merging with the auxiliary texture, and normalization are performed after each iteration.

### 4.8 Results

All numerical images were performed on a PC with graphic card NVIDIA GTX 680 and implemented in CUDA. We used the same color scheme to render all continuous representations of projections, see Figure 4.5 (e). The texture size used for computation is \(1024^2\) pixels. We reserved 10\% for the border, such that the size of the rendered texture is \(921^2\) pixels. Computation of the whole spectral representation requires \(1024^2/4096 = 256\) iterations, where parameter 4096 is chosen to achieve a high occupancy of the GPU. If other is not specified, \(s_{\text{overall}}(x) = \text{const}\) is used. Other information about parameters and datasets is shown in Table 4.1.
Table 4.1: Parameters and thresholds for all experiments. Shown are values of the global smoothing parameter $k$, the threshold that defines which splats are labeled as small, the size of dataset, and the number of splats computed by the spectral method for the given threshold.

<table>
<thead>
<tr>
<th>dataset</th>
<th>$k$</th>
<th>threshold</th>
<th>total number of points</th>
<th>number of large splats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bucky Ball</td>
<td>2.0</td>
<td>$0.02 \cdot R_u$</td>
<td>32768</td>
<td>13986</td>
</tr>
<tr>
<td>White Dwarfs</td>
<td>1.6</td>
<td>$0.008 \cdot R_u$</td>
<td>24149</td>
<td>21676</td>
</tr>
<tr>
<td>Tornado</td>
<td>0.6</td>
<td>$0.01 \cdot R_u$</td>
<td>262144</td>
<td>82856</td>
</tr>
</tbody>
</table>

Figure 4.6: Continuous scatterplots of merging White Dwarfs dataset. When all footprints are progressively computed using spectral method, small splats cause noise, which is visible at first iterations and disappears later. After all iterations are completed, the result is identical to one obtained by the direct approach, i.e., when all elliptical footprints are blended as textures.
4. PROGRESSIVE RENDERING OF CONTINUOUS SCATTERPLOTS

Figure 4.7: Hybrid approach applied to the merging White Dwarfs dataset.

First, we demonstrate the convergence of the pure spectral approach. The dataset represents an astrophysical simulation of two merging White Dwarfs. The simulation was executed by means of the Smoothed Particle Hydrodynamics method [69, 70]. The data includes unstructured nodes’ positions $x_i$, lengths $h_i$, several scalar attributes and their gradients. Dimensions of the scatterplot are internal energy and temperature fields in horizontal and vertical directions, correspondingly. Results for $s_{\text{overall}}(x)$ to be equal to the density field are shown in Figure 4.6. Small splats have poor spectral representation and, thus, serve as a source of noise. This noise is eliminated during progressive computation of Fourier magnitudes. After all iterations are completed, the result is identical to one obtained by the direct method. When the hybrid approach is applied to the same dataset, the initial level of noise is much lower, see Figure 4.7.
4.8. RESULTS

Next, we used the “Tornado” dataset, courtesy of [68]. It is given as an analytical function describing a velocity profile in a volume. We sampled the velocity field at $64^3$ random uniformly distributed locations. Results obtained by the hybrid approach are presented in Figure 4.8. Velocity components in $x$ and $y$ directions serve as the two axes of the scatterplots. We intentionally chose a relatively small $k$ to demonstrate the effect of detailization during progressive rendering. Small global smoothness makes the sizes of footprints small, thus, even some individual splats remain distinguishable in the continuous scatterplot. The first iteration of the hybrid approach results in a slightly smeared image with more and more details appearing at later steps.

Computational times are listed in Table 4.2. Both the spectral and the hybrid approaches deliver preliminary results much earlier than the direct splatting method. Though, the first iteration of the hybrid method takes a longer time than one iteration of the spectral method, its later iterations are faster and the quality of the result is higher.

Figure 4.8: Hybrid approach applied to the “Tornado” dataset.
Table 4.2: Computation times for construction of continuous scatterplots using direct approach, pure spectral representation and the hybrid algorithm (all methods are implemented on GPU).

<table>
<thead>
<tr>
<th>dataset</th>
<th>direct, per iter.</th>
<th>spectral, 1st iter.</th>
<th>hybrid, 1st iter.</th>
<th>hybrid, from 2nd iter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bucky Ball</td>
<td>3842ms</td>
<td>119ms</td>
<td>435ms</td>
<td>48ms</td>
</tr>
<tr>
<td>White Dwarfs</td>
<td>895ms</td>
<td>95ms</td>
<td>269ms</td>
<td>76ms</td>
</tr>
<tr>
<td>Tornado</td>
<td>1254ms</td>
<td>591ms</td>
<td>282ms</td>
<td>203ms</td>
</tr>
</tbody>
</table>

To measure the error of the proposed methods at $j$-th iteration, we computed the root-mean-square error

$$L_j = \left( \frac{1}{R_{ul} \cdot R_{vr}} \sum_{p_x=1}^{R_u} \sum_{p_y=1}^{R_v} (\bar{\sigma}(p_x, p_y) - \bar{\sigma}_{direct}(p_x, p_y))^2 \right)^{1/2},$$

where $p_x$ and $p_y$ stands for pixel indices, $\bar{\sigma}_{direct}$ is the density computed by the direct approach, $\bar{\sigma}$ can be computed either by the spectral or by the hybrid method, and the bars denote that the densities are normalized (individually). Results are shown in Figure 4.9. It is evident that the error is gradually improved along the progressive computations. The error of the hybrid method after first iteration is significantly less than the analogous error of the spectral one. Moreover, only a few first iterations of the hybrid method suffice to closely approach a stable state.

The hybrid approach has lower error, but the first step also needs more computation time. As follows from the computation times in Table 4.2, almost 7 spectral iterations can be done before the first iteration of the hybrid method is completed. However, the initial error of the hybrid approach is lower than the error of the spectral approach after 7 iterations and decreases faster in the consequent steps. This holds for all datasets. By the time, when the first result of the hybrid approach is delivered, only 11% (“Bucky Ball”), 30% (“two White Dwarfs”) and 22% (“Tornado”) of samples can be processed by the direct method, which is insufficient for performing a reliable analysis.
4.8. RESULTS

Figure 4.9: Error behavior along the iterative construction of continuous scatterplots. The plots present the root-mean-square errors of the spectral algorithm and the hybrid method. Results for the "Bucky Ball" (red), two White Dwarfs (blue) and the "Tornado" (magenta) datasets are shown. The error of the hybrid approach applied to the "Bucky Ball" dataset (red line in (b)) is multiplied by a factor of $10^3$ to be visible.

Figure 4.10: Plots of calculation times of appearance of the first meaningful image against output resolution (pixels) computed for "Bucky Ball" dataset. Results for the direct splatting (red), the hybrid method (blue) and the spectral approach (green) are shown.

The benefit of our approach becomes extremely significant when producing high-resolution outputs. Figure 4.10 demonstrates the dependence of calculation times for the first meaningful picture on the output resolution. In particular, the speed-up of 500 times is achieved for $R_u = 3686$ pixels. Note that the values of time (vertical axis) are scaled logarithmically. Non-monotonic behavior of the green line is related to higher efficiency of FFT when $R$ is a power of 2.
4.9 Conclusion

We have seen that when producing high-resolution continuous scatterplots, when dealing with highly adaptive sampling in physical space, when zooming into a region of interest on a scatterplot, or when increasing the global smoothness of the result, sizes of some footprints become large up to the range of the screen size. In such situation, a direct accumulation of the splats in the final image gets extremely slow and significantly affects the overall performance, which hinders interactivity. Our method allows overcoming this drawback by the effective use of the spectral representation of large footprints.

The proposed progressive rendering approach fulfills all three requirements formulated in the Introduction section. First, small splats are directly aggregated into the density texture at the very first step of the algorithm. These splats usually have highest density values and thus affect the overall picture most, since they mostly determine the range of the transfer function being applied. Second, due to decay of the Fourier coefficients of a smooth function, each next iteration in the Fourier domain will have less impact than the previous step. Thus, changes in the displayed picture lessen over the iteration steps and stabilize to the final state quite rapidly, which was shown in a number of tests. Finally, only missing modes of spectral representation are added in each subsequent step, so that no re-computation of any part of earlier obtained results is needed.

We have been able to significantly speed up the appearance of the first result of the hybrid method when compared to the full-resolution direct splatting approach. In our tests, the speed-up varied from 3.3 times to two orders of magnitude depending on the resolution of the output. This allows for computation times suitable for interactive analysis, e.g., in scatterplot matrices.
Chapter 5

Continuous representation of projected attribute spaces

This chapter is based on the publication:


For the visual analysis of multidimensional data, dimension reduction methods are commonly used to project to a lower-dimensional visual space. In the context of multifields, i.e., volume data with a multidimensional attribute space, the spatial arrangement of the samples in the volumetric domain can be exploited to generate a Continuous Representation of the Projected Attribute Space (CoRPAS). Here, the sample locations in the volumetric domain may be arranged in a structured or unstructured way and may or may not be connected by a grid or a mesh. We propose an approach to generate CoRPAS for any sample arrangement using an isotropic density function. An interactive visual exploration system with three coordinated views of volume visualization, CoRPAS, and an interaction widget based on star coordinates is presented. The star-coordinates widget provides an intuitive means for the user to change the projection matrix. The coordinated views allow for feature selection in form of brushing and linking. The approach is applied to both synthetic data and data resulting from numerical simulations of physical
5.1. INTRODUCTION

phenomena. In particular, simulations based on Smoothed Particle Hydrodynamics are addressed, where the simulation kernel can be used to produce a CoRPAS that is consistent with the simulation. We also show how a logarithmic scaling of attribute values in CoRPAS is supported, which is of high practical relevance.

The concept of the continuous representation of projected attribute spaces, as well as the star-coordinates widget, were developed and introduced by Vladimir Molchanov. Development and implementation of the computational algorithms using GPU for the rendering of continuous scatterplots in both linear and logarithmic scales were done by Alexey Fofonov. This constitutes a contribution in frames of the thesis. Successful combination of the methodological base and efficient usage of the computational resources allows for fast and interactive visual analysis of the attribute space.

5.1 Introduction

When acquiring volume data by taking measurements, one samples a mostly continuous data field at discrete spatial locations. Similarly, in numerical simulations, one models continuous phenomena and discretizes them for computational purposes. Because of the continuity assumption, it is common practice to reconstruct a continuous field from the values at the discrete samples for visualization purposes.

Often, the measured or simulated volume data consist of multiple fields, i.e., they represent a volumetric dataset with a multidimensional attribute space. In the multidimensional attribute space, each sample is represented as a point in an \( m \)-dimensional space, where \( m \) is the number of attributes. To visualize the distribution of the samples in the attribute space, many different techniques have been proposed in the context of non-spatial data visualization. Common examples are scatterplot matrices, parallel coordinates, or 2D scatterplots after dimension reduction. Dimension reduction algorithms such as projections try to create a similarity-based layout of the \( m \)-dimensional points in the two-dimensional visual space with respect to some optimization goal.

When considering multifields, the reconstruction of continuous fields from discrete samples can also be applied to the multidimensional attribute space, where interpolation
techniques are applied to those points that reflect a neighborhood relationship in volume space. This consideration has recently led to visualization approaches of continuous scatterplots and continuous parallel coordinates [50–52]. In terms of a multifield data visualization, scatterplots are using projections orthogonal to two given attribute axes in the \( m \)-dimensional space. When restricting oneself to such representations, multifield visualization requires one to deploy entire scatterplot matrices to capture the entire dataset. Scatterplot matrices are rather tedious to explore interactively when dimensionality grows. Already for ten-dimensional data, it is highly inefficient to investigate the 45 plots. Also, restricting oneself to orthogonal views only, certain phenomena like clusters may be missed. In our work, we propose a method to construct Continuous Representations of Projected Attribute Spaces (CoRPAS) of multifields for arbitrary subsets.

There exists a large range of projection methods that automatically compute the best projection with respect to some design goal. However, design goals such as cluster segregation and distance preservation often contradict each other so that the design of a projection that is best for all purposes is impossible. Instead, our system provides the user with means for an interactive and intuitive manipulation of the projection using a star-coordinate widget. This approach is feasible, as the number of attributes is typically not higher than 10 to 20.

In terms of the structure of the underlying multifield, our approach supports any spatial sampling. The samples of the volume (or physical) space can build a regular pattern or be unstructured. The former case is typical for grid-based numerical methods and medical imaging techniques. The latter is common for sensor data, simulations over unstructured meshes, and particle-based simulations. If the data are given on a grid or mesh, the connectivity is used for the reconstruction of the continuous field by standard interpolation methods like (tri-)linear interpolation. In case of no given connectivity between the samples, scattered data interpolation techniques like the ones based on inverse distances, radial bases, natural neighbors, or Moving Least Squares (MLS) can be applied.

Our system supports several types of interpolation kernels. In particular, if the data source is a simulation, the interpolation method used in the simulation is the best choice
for a continuous visualization of attribute spaces. In our work, we apply the proposed method to Smoothed Particle Hydrodynamics (SPH) simulation data. The continuous reconstruction of the data fields is performed using the SPH kernel function, usually a cubic spline. For that purpose, we compute the footprint of the kernel function after projection to the visualization domain. Since for many simulations it may be beneficial to use logarithmic scaling, we also extend our approach to computations of footprints for logarithmic plots.

The main contributions of our paper are:

- Continuous scatterplots are generalized to continuous representations of multi-dimensional attribute spaces ($m > 2$) for any linear projections.
- The proposed method works on any spatial sampling of the volumetric domain including unstructured point-based data. It requires estimation of attribute gradient fields and associated volumes at samples to compute geometrical properties of their footprints in the attribute space.
- An algorithm for the construction of continuous logarithmic plots is proposed.

Besides that, a visualization system for the interactive visual analysis of multifields was developed. It allows the user to intuitively and interactively change the projection matrix for which we adopted the star-coordinate widget used by Teoh and Ma [71] and Bordingon et al. [72]. The system supports coordinated views between physical and attribute space visualizations and allows for brushing and linking to detect features. Moreover, a method for data analysis by changing the weighting function is discussed. Finally, the proposed techniques are applied to various unstructured data, in particular, to SPH simulation data, where the simulation kernel can be used for consistent CoRPAS.

For the related work about multi-field data analysis we refer to Chapter 2.

In our work we are exploiting the continuity property of multifield data to generate continuous representations of the projected attributed spaces and propose a fully interactive approach that does not rely on the success of automatic components.
5. CONTINUOUS REPRESENTATION OF PROJECTED ATTRIBUTE SPACES

5.2 Background

5.2.1 Physical, Attribute, and Visual Domains

Let the relationship between the spatial (or physical) domain \( X \subseteq \mathbb{R}^n \) and the attribute domain \( Q \subseteq \mathbb{R}^m \) be given by a mapping \( \tau : \tau(X) = Q \). Then, data sampled at positions \( \{x_i\} \subseteq V \) have attributes \( \{q_i = \tau(x_i)\} \). We do not assume any a priori connectivity information regarding positions of samples \( x_i \), i.e., the samples’ distribution can be arbitrary. Most practical applications deal with settings of two or three spatial dimensions. In the following, we concentrate our consideration on case \( n = 3 \), i.e., volumetric data. The planar case \( n = 2 \) can be trivially deduced. The attribute space \( Q \) is linearly projected to the visual domain \( U \subseteq \mathbb{R}^d \): \( P \cdot q_i = u_i \) for all \( i \). In practice, the projected data are shown on a screen so that the visual domain has \( d = 2 \). Figure 5.1 illustrates our notations.

Mapping the multidimensional attribute values given at samples to the visual domain results in a discrete projection. The actual positions of the individual samples in the physical space play no role, so, this valuable information is ignored when projecting. CoRPAS can help the user to get a better understanding of attribute distribution in the data in connection with spatial information. The key idea of the method is to map a sample together with its small neighborhood rather than a single point. The collection of such neighborhoods equipped with local spatial densities \( s_i(x) \) covers the entire volume \( X \) with possibly overlapping subvolumes. We call the image of \( i \)-th neighborhood a footprint \( \sigma_i(u) \).

Figure 5.1: Physical space \( X \) is mapped by \( \tau \) to the attribute space \( Q \), which is then projected to the visual domain \( U \) by linear operator \( P \). A spatial density \( s_i(x) \) is defined in a spherical neighborhood of sample \( x_i \), which has an elliptical footprint in \( U \). Scalar function \( \sigma_i \) stands for density on the footprint.
Figure 5.2: Discrete scatterplot (a) may correspond to different spatial configurations, e.g., the ones shown in (b), (c), (d). Continuous scatterplot (e) exhibits smooth transitions between spatially adjacent clusters, i.e., it can only correspond to (d).

Figure 5.2 shows a toy example of a discrete scatterplot. The scatterplot may correspond to different spatial configurations, as it does not provide any information about which of the identified clusters have common interfaces in the physical space. The continuous scatterplot encodes information about spatial coherence of clusters in the form of smooth transitions between spatially adjacent clusters.

A continuous representation is defined as the limit of discrete representations when the number of samples increases unboundedly. Assuming that the mapping $\tau$ does not change the number of points [51], the mass accumulated in the physical domain should be equal to the mass in the visual domain. Thus, construction of CoRPAS is related to finding an attribute space density function $\sigma_{\text{overall}} : \mathbb{R}^d \to \mathbb{R}$, $\sigma_{\text{overall}}(u) = \sum \sigma_i(u)$, knowing the spatial density function $s_{\text{overall}} : \mathbb{R}^n \to \mathbb{R}$, $s_{\text{overall}}(x) = \sum s_i(x)$, such that

$$\int_{X_0} s_{\text{overall}}(x) \, d^n x = \int_{U_0} \sigma_{\text{overall}}(u) \, d^d u,$$  

(5.1)

for any $X_0 \subset X$ and $U_0 = \tau(X_0)$. This condition uniquely defines $\sigma_{\text{overall}}$. Equation (5.1) with $X_0 \equiv X$ and $U_0 \equiv U$ is called total mass conservation.
5.2.2 Spatially Unstructured Data

To deal with unstructured point-based data, we apply scattered data interpolation techniques. MLS is such a method, which can approximate both values of the sampled field and its derivatives of arbitrary order [16, 73–75]. The value \( f(y) \) at any position \( y \) is found as the solution of the minimization problem:

\[
f(y) = \sum_k c_k g_k(y),
\]
\[
\{c_k\} = \arg\min_i \sum \omega(y - x_i, h)(f(x_i) - f_i)^2,
\]

where \( g_k \) denote some basis functions (e.g., polynomials), \( \omega \) is a weighting function having support size (or characteristic length) \( h \), and \( f_i \) are the given function values at \( x_i \).

It is well-known that the smoothness of the MLS approximation is determined by the smoothness of the weighting function. Thus, for sufficiently smooth \( \omega \), MLS approximation can also be used to estimate derivatives of the field. If \( \omega \) has finite support, only few neighbor samples \( x_i \) contribute to the value at \( y \). For each sample a list of nearest neighbors can be precomputed. The search of neighbors can be effectively performed using a \( kd \)-tree [76].

SPH is a simulation technique that produces unstructured point-based data. It was proposed by Gingold and Monaghan [70] and Lucy [69] and is currently widely used in astrophysics to simulate evolution of objects like stars. Samples (or particles) evolve in time and do not maintain any connected structure during simulation. They provide a discrete representation of an object and carry a number of scalar (mass, density, temperature, etc.) and vector (velocity, force, etc.) values. The method relies on an interpolation of continuous fields from disconnected particle locations using non-negative radial reconstruction kernels.

Each particle with index \( i \) has its own spatial range of influence, which is referred to as smoothing length \( h_i \). The contribution of a single particle to the overall continuous field \( f(x) \) is then defined by a smoothing kernel \( s_i \) with its support proportional to \( h_i \).
The contributions of all particles deliver an SPH approximation to \( f(x) \):

\[
f(x) \approx \sum_i f_i \frac{m_i}{\rho_i h_i^n} s_i(x - x_i),
\]

(5.2)

where \( m_i, \rho_i, \) and \( f_i \) are mass, density, and function value of the particle with index \( i \), respectively. Thus, \( f(x) \) is locally represented as a superposition of overlapping kernels. Usually, the number of locally contributing samples is about 100. Provided the kernel functions \( s_i \) are smooth, derivatives of \( f(x) \) can be approximated by differentiation of Equation (5.2).

Standard SPH basis functions \( s_i \) are radial. In the following we use the notation \( f(x) = f'(\|x\|) \) when radiality of function \( f \) is to be emphasized. The most widely used of all SPH kernel functions are the Schoenberg B-splines [77], in particular, the cubic spline \( s'_i(t) = s'(t/h_i) \), where

\[
s'(q) = C \begin{cases} 
(2 - q)^3 \big/ 4 - (1 - q)^3, & \text{if } 0 \leq q < 1, \\
(2 - q)^3 \big/ 4, & \text{if } 1 \leq q < 2, \\
0, & \text{if } 2 \leq q
\end{cases}
\]

and constant \( C \) depends on dimensionality \( n \).

### 5.3 CoRPAS

Continuous multifields in physical domain can be reconstructed from the values at the samples via some interpolation. CoRPASs serve to represent the reconstructed fields. To do so, the physical domain is decomposed into parts which are mapped to the visual space and accumulated taking into account relation (5.1). Some simplifications and assumptions concerning the decomposition geometry and local field behavior are needed to balance the accuracy and performance of the mapping. These aspects define the key differences of existing approaches.
Bachthaler et al. [51] focus on tetrahedral meshes with linear interpolation using barycentric coordinates. The spatial density is taken to be constant within each tetrahedron. The footprint of a tetrahedron is represented by up to four triangles with linearly varying density $\sigma$. In the follow-up paper [52], the authors work with rectangular grids and use the original interpolation method to iteratively subdivide the volume into small rectangular subvoxels until their footprint sizes reach a threshold. The threshold is used to balance the trade-off between performance and accuracy. Then, all subcells are mapped to the scatterplot as polygons with constant intensity $\sigma$ to fulfill Equation (5.1). Both methods use the underlying data structure to decompose the volume into non-intersecting tetrahedra or cuboids. Thus, they perfectly partition the physical domain in terms of the overall density function fulfilling $s_{\text{overall}}(x) = \text{const}$. However, the former approach is limited to tetrahedral meshes, while for the latter approach Equation (5.1) does not hold for an arbitrary subvolume $X_0$ and performance highly depends on the non-intuitive threshold value.

In the recent approach proposed by Heinrich et al. [50], the volume is represented as a collection of overlapping spherical regions centered at samples. The overall density in physical space is given by

$$s_{\text{overall}}(x) = \sum_i w_i s_i(x),$$

(5.3)

where $w_i$ are scalar weights and the local density functions $s_i$ centered at $x_i$ are taken to be Gaussian kernels (truncated in practice), such that the overlapping parts of spheres get smaller weights. The spatial spheres have elliptical footprints with Gaussian density. The geometry of each footprint is determined by the linear part of $\tau$. Advantages of the method include: Elliptical splats are easy to blend, Equation (5.1) is satisfied for a locally linearized mapping $\tau$, and the user can easily control the global smoothness of the continuous scatterplot by proportional scaling of the spheres’ radii (by adjusting a global parameter $k$). However, the approach is still limited to orthogonal projection of the attribute space (in form of scatterplots), to linearly scaled plots, and to $s_{\text{overall}}(x) = \text{const}$. Besides that, a single Gaussian kernel is considered in the paper, which limits the application of the method to regularly sampled data. In the general case, parameters of the spatial kernels and their footprints have to vary from sample to sample depending on the local sample density. Our work serves to overcome the limitations above.
Several approaches can be proposed to produce a continuous scatterplot for unstructured spatial data. We discard all but one approach based on the following observations. (1) Regularization of data via interpolation to an auxiliary grid introduces errors, especially if the original samples’ distribution is highly irregular. (2) Partitioning the volume into non-overlapping regions is possible in the form of Voronoi tesselation. However, performance of such an approach is expected to be low due to the complicated decomposition step and the need to map of arbitrarily-shaped Voronoi cells for blending. The use of isotropic kernels is then the only natural choice approved by many computational methods like SPH and MLS. This motivates us to use radial kernel functions of the form $s(x) = s^r(\|x\|)$.

In the following, we will detail the individual steps for generating a CoRPAS of multifields. Our approach does not consider any constraints on the spatial locations of the samples in volume space, nor does it assume any connectivity. Using the notations introduced in the previous section, the main steps of the algorithm can be outlined as follows:

1. For every sample, estimate the radius of sample’s spherical neighborhood, at which a local linearization of the mapping $\tau$ is found, and approximate the gradients of all attribute fields at the sample position.
2. For every neighborhood, compute its elliptical footprint in the visual domain.
3. Find the overall density distribution by summing up the contributions from all footprints.
4. Apply a proper transfer function to visualize the result.

### 5.3.1 User-defined Projection

Reducing the dimensionality of high-dimensional data to a lower-dimensional space has been subject to research for many decades. Most commonly used are classical dimension reduction approaches like Principal Component Analysis (PCA) [78] or Multidimensional Scaling approaches (MDS) [79] that try to find a mapping that preserves distances, or force-directed placements. To not insert too much distortion and keep computation costs low, linear projections are often preferred. However, optimization
goals may be contradicting, as one is often interested in preserving distances as much as possible and, at the same time, would like to keep clusters (groups of similar attribute values) separated during projection. Contradicting goals make it impossible to find one projection which is always best. Moreover, one projection, in general, does not suffice to convey all (and not even most) important features. Therefore, it is desirable to have interactive means to analyze the multidimensional attribute space.

Our approach provides such means in form of an intuitive star-coordinates interaction widget. Star coordinates [80,81] arrange the coordinate axes of a multidimensional space on a circle placed on a two-dimensional plane with axes having their origin at the center of the circle and an arrangement exposing equal angles between adjacent axes. A multidimensional point is mapped to a two-dimensional point by summing the unit vectors of each coordinate multiplied with the coordinate of the multidimensional point. Teoh and Ma [71] extended this concept by not imposing any constraints on the coordinate axes (neither length nor direction) except for being rooted in the origin and by allowing for interactive change of the coordinate axes. Our interaction widget follows the same idea.

Any linear projection of the attribute space $Q$ onto the visual space $U$ can be expressed in the form $\mathbf{u} = P \cdot \mathbf{q}$ with some $m \times d$ matrix $P$. The columns of $P$ are the images in $U$ of basis vectors in $Q$. Thus, columns of $P$ can be visualized as basis vectors of a star-coordinates system. Initially, all basis vectors (which start in the origin) are distributed uniformly over the unite circle as shown in Figure 5.3 (a), where the colors of the axes are chosen from a categorical color map to have most different hues. This default state can be restored any time by just one mouse click. Our system allows the user to interactively manipulate the projection by means of changing the positions of the end-points of basis vectors. Entries of $P$ are then recomputed and used to project the multifield data. Thus, any linear projection has its unique representation in the widget and any configuration of basis vectors can be interpreted as matrix $P$. In particular, every vector can be deactivated by placing its end-point at the origin. Then, the respective column of projection matrix has only zero entries and the disabled attribute field does not affect the final projection. Thus, our approach naturally and intuitively includes the feature of analyzing attribute subspaces data (including the detection of clusters in subspaces).
5.3. CORPAS

Figure 5.3: Continuous representation of projected 5D attribute space of Smoothed Particle Hydrodynamics simulation data (a disrupting white dwarf). Projection matrices according to (a) and (c) result in continuous representations of projected attribute spaces (b) and (d), correspondingly. Manipulation of projection matrix by means of star-coordinates widget allows the user to visually separate two clusters of particles representing the star’s core and the tail. These clusters are interactively selected in (d) and the locations of the particles in physical domain are shown (e). The color map used in (b) and (d) is presented in (f).

5.3.2 Kernel Functions and Their Supports

Isotropic kernels have the nice property that they have elliptical footprints under linear transformations. The computation of footprint density can be done very accurately, in some cases even analytically (Gaussian kernel [50]). For practical reasons, shape function s should have a compact support or, at least, be a fast decaying function like a Gaussian. We wish to emphasize that even if the kernel is a globally supported function, the smoothing length $h$ has still a meaning: In this case, $h$ is a scaling parameter defining the decay rate of the kernel. As shortly discussed by Heinrich et al. [50], large $h$ leads to smeared results, whereas vanishing $h$ corresponds to discrete projection. In the general case, support size $h_i$ varies from sample to sample. Heinrich et al. [50] considered only one kernel, thus, avoided the problem of optimal choice of individual support sizes.

We make the following observations: If samples are placed at the nodes of a regular grid, $h_i$ can be derived from the geometrical parameters of the cells. When visualizing SPH data, the smoothing lengths of particles stem from the simulation and are provided with the data. For general unstructured point-based data, we propose to choose $h_i$ according to the local sample density. In practice, we set $h_i$ proportional to the distance to the $j$-th nearest neighbor with $j$ defined by the user. In our experiments we used $j = 40$ and a linear MLS ansatz function to estimate missing gradients of the fields. Similar to
the approach by Heinrich et al. [50], the user can scale all $h_i$ by a global factor $k$, which allows to control the global smoothness of CoRPAS.

### 5.3.3 Footprint Computation

We substitute the mapping $\tau$ by its linearization (Taylor’s series truncated after its second term) in a vicinity of a considered sample point $x_i$ leading to

$$q \approx \tau(x_i) + D_\tau \cdot (x - x_i), \quad D_\tau = (\nabla_x q_1(x_i), \ldots, \nabla_x q_m(x_i)),$$

where $D_\tau$ represents the non-constant part of the mapping and $x = \tau^{-1}(q)$ is a point in a neighborhood of the sample.

Since the user-defined projection $P$ is also linear, we obtain

$$u = P \cdot q = P \cdot \tau(x_i) + P \cdot D_\tau \cdot (x - x_i).$$

Let $u_0 = P \cdot \tau(x_i)$ and $T = P \cdot D_\tau$, where $T = (n_1, \ldots, n_d)^T$ is a $d \times n$ matrix. Thus, we get the following relation between physical and visual domains: $u - u_i = T \cdot (x - x_i)$. We wish to emphasize that this procedure is performed for $x$ within a small spherical neighborhood of radius $k h_i$ of $x_i$.

At this stage, we reduced the problem to the situation discussed in [50] with a re-defined linear transformation. To complete the exposition, we reproduce the formulae from [50] describing the geometry of an elliptical footprint and correct a misprint therein. The formulae are designed for the case $n = 3$, arbitrary $m$, and $d = 2$, i.e., when the volumetric multifield data is projected onto a two-dimensional screen. Here, $T = (n_1, n_2)^T$ is assumed to be non-singular, i.e., vectors $n_1$ and $n_2$ are linearly independent. According to [65], the extents $S_{i,x}$, $S_{i,y}$, and the rotation angle $\theta_i$ of the screen-space ellipse are defined as

$$S_{i,x} = kh_i \sqrt{\lambda_1}, \quad S_{i,y} = kh_i \sqrt{\lambda_2}, \quad \cos \theta_i = \frac{b}{\sqrt{b^2 + (\lambda_1 - a)^2}},$$
where $kh_i$ is the radius of the sphere in physical domain,

$$\lambda_1 = \frac{a + c + e}{2}, \quad \lambda_2 = \frac{a + c - e}{2}, \quad e = \sqrt{(a - c)^2 + 4b^2},$$

$$a = n_1(x_i) \cdot n_1(x_i), \quad b = n_1(x_i) \cdot n_2(x_i), \quad c = n_2(x_i) \cdot n_2(x_i),$$

and $k$ is a global parameter controlling the smoothness of CoRPAS. Now, every footprint can be obtained from a template texture by its axial scaling with coefficients $S_{i,x}$ and $S_{i,y}$, rotation with angle $\theta_i$, and translation by vector $u_i$. Note that in [50], the angle $\theta$ implicitly depends on $k$ as it depends on $\lambda_1$ in notations therein. This misprint is corrected in our formulae above.

After the shape of the footprint is defined, the task is to find the density function $\sigma_i$. Since $s_i$ are scaled versions of the template function $s$, the same is true for the respective footprints $\sigma_i$ and the template $\sigma$. We set

$$\sigma'(r) = \int_{-\infty}^{\infty} s'\left(\sqrt{\rho^2 + x_3^2}\right) \, dx_3, \quad \rho^2 = x_1^2 + x_2^2.$$

Here, we effectively use the isotropy property of $s$: The integral of $s$ is the same independent of the integration direction. We picked to integrate in direction $x_3$, but could have also chosen $x_1$ or $x_2$ instead.

To compute $\sigma_i(u)$, the template $\sigma(u)$ is multiplied with coefficient $A_i = w_i h_i^n S_{i,x}^{-1} S_{i,y}^{-1}$ and transformed with parameters $S_{i,x}, S_{i,y}, \theta_i$, and $u_i$ to fit the geometry described above. To show the correctness of $A_i$, it is enough to check that the mass conservation holds. Note that rotation and translation do not affect the accumulated mass, thus, we take $\theta_i = 0$ and $u_i = (0, 0)$ without loss of generality. Then,

$$\int_Q \sigma_i(u) \, d^u = w_i h_i^n \int_Q \sigma(v) \, d^v$$

$$= c w_i h_i^n \int_X s(y) \, d^y = c w_i \int_X s_i(x) \, d^x.$$  

Summation over index $i$ gives then Equation (5.1). Coefficient $c$ does not play a role, since its effect is eliminated when applying the transfer function.
The choice of local kernels $s_i$ does not affect the quality of the result if sufficiently smooth functions are used. Gaussian kernels need to be truncated, while B-splines have compact support by definition. We note, however, that global smoothing parameter $k$ should be adapted for consistency when switching between different kernels.

A principle choice is made when defining weights $w_i$ in Equation (5.3). This aspect is not discussed in [50] because of two reasons: First, $s_{\text{overall}}(x) = \text{const}$ was the target for the datasets used in that paper, second, for regularly structured data $w_i = \text{const}$ for all $i$ is the obvious choice. At this point it is useful to compare Equation (5.3) with Equation (5.2). Setting $f(x) = 1$ and $f_i = 1$, we derive an approximate decomposition of the volume

$$1 \approx \sum_i \frac{m_i}{\rho_i h_i^3} s_i(x),$$

where, physically, both $m_i/\rho_i$ and $h_i^3$ are volumes associated with samples. In case when mass and density values are not known for samples, we set $w_i = 1$ as it is done in [50]. However, when processing physical simulation data (like SPH), this choice may impede the actual analysis task. The samples representing background (low-density, low-energy, low-temperature region), which may occupy the largest part of the volume under consideration, would be weighted by large coefficients $w_i$ and would appear as bright spots in CoRPAS. This region is of no practical relevance, since the investigator would want to focus on the analysis of the modeled object and not on the surrounding environment. Thus, other weighting schemes are required.

Setting $f(x) = \rho(x)$ and $f_i = \rho_i$, we obtain

$$\rho(x) \approx \sum_i \frac{m_i}{h_i^3} s_i(x),$$

which means that $w_i = m_i/h_i^3 \sim \rho_i$. Here, the background plays only a minor role, such that the massive parts of the object can be explored. Analogously, weights can be chosen with respect to energy, temperature, and any other positive functions according to the phenomenon being studied.
Figure 5.4: Comparison of footprints in linear (left) and logarithmic (center and right) plots. All footprints parameters are the same except for the position of center $u_i$. Several level curves of density function $\sigma$ are shown.

### 5.3.4 Construction of Logarithmic Plots

Many scientific datasets have attribute distributions that lead to clutter when displayed on linearly scaled plots. This problem can be alleviated by the use of logarithmic plots, which is a very common approach in visual analyses of scientific data. In the context of CoRPAS, we want to extend our approach by using footprints in the visual space that is scaled logarithmically. In this section, we propose an algorithm for two-dimensional continuous scatterplots, i.e., projections into the space spanned by two of the given dimensions, where two respective attributes vary logarithmically along the axes.

A naive approach would be to compute the logarithm of the given data values (assumed to be strictly positive) in a pre-processing step and then apply our techniques as described above. However, this would be inconsistent with the local linearization of $\tau$. To illustrate the error introduced, we show respective elliptical footprints with linear and logarithmic scaling in Figure 5.4. The linearly scaled footprint is symmetric along its major axes with respect to the sample image $u_i$. The logarithmically scaled footprints do not possess this property. Since our approach produces a continuous representation by summation of elliptical footprints after affine transformation, the use of logarithmically transformed data would result in significant errors.

For the solution we propose instead, we first note that the template $\sigma$ is a radial function by construction. Thus, its values can be effectively stored in a single 1D look-up table. Second, we need to derive an analytical form of the elliptical footprint. The
support of the footprint (before translation by vector \( \mathbf{u}_i \)) is given by

\[
E_i(\mathbf{v}(\mathbf{u})) = \frac{1}{2k^2h_i^2} \left( \frac{(v_1 + v_2)^2}{1 + \cos \theta_i} + \frac{(v_1 - v_2)^2}{1 - \cos \theta_i} \right) \leq 1 \tag{5.4}
\]

with \( v_i = u_i / \|\mathbf{u}_i\| \), \( i = 1, 2 \). It is easy to prove that this form is equivalent to our description above by showing that the axes of ellipse \( E_i(\mathbf{v}(\mathbf{u})) = 1 \) coincide with \( S_{i,x} \) and \( S_{i,y} \).

For the construction of logarithmic plots, we first map the considered sample to the visual domain, \( \mathbf{x}_i \rightarrow \mathbf{u}_i \), and estimate the bounding box of the footprint. Then, we process the pixels confined by the bounding box as follows:

1. For each pixel, compute the attribute values \( \mathbf{u} \) by inverting the logarithmic scaling.
2. Compute \( E_i(\mathbf{v}(\mathbf{u} - \mathbf{u}_i)) \).
3. If inequality (5.4) holds, i.e., the pixel belongs to the footprint’s support, compute density \( \sigma^r_i(\sqrt{E_i}) \), where the argument of \( \sigma^r_i \) is the algebraic distance to the footprint’s boundary (5.4). The square-root operation can be practically avoided if values of \( \sigma^r_i(t^2) \) are stored in the look-up table. Add the computed value to the resulting plot.

Iterating over all samples as described above, results in a continuous density \( \sigma_{\text{overall}}(\mathbf{u}) \). We note that the mass conservation (5.1) holds if logarithmic scaling of \( \mathbf{u} \) is taken into account when integrating over \( U_0 \).

### 5.4 Results

For our results we used a cubic spline as spatial density function in case of SPH data and a Gaussian kernel otherwise.

To validate that our approach is applicable over any spatial sampling, we compared CoRPAS of regular and unstructured data. We sampled the “Bucky Ball” dataset (regular grid, \( 32^3 \) nodes, courtesy of The Volume Library) at uniformly distributed \( 32^3 \) random points using trilinear interpolation. Continuous scatterplots that map data values
5.4. RESULTS

Figure 5.5: CoRPAS of “Bucky Ball” dataset: Original data value in horizontal axis and gradient magnitude in vertical one. Gridded (left) and uniform random data (right) contain the same number of points and result in similar plots.

(horizonal axis) against gradient magnitude (vertical axis) are presented in Figure 5.5. Gradients are computed using a finite differencing scheme in the regular case and are based on MLS in the unstructured case. The MLS computation has an effect of smoothing, which explains the slightly lower height of the arc. However, we can note that the plots have qualitatively similar structure in terms of geometry and density.

Figure 5.3 illustrates how our system can be used to interactively explore multifield data based on CoRPAS and coordinated views. The star-coordinate widget allows the user to interactively manipulate the projection matrix, which results in different projections. In the example, we demonstrate how one can interactively detect clusters corresponding to the core and the tail of a simulated white dwarf. The presented SPH data consists of 0.5M particles and has five scalar attributes: Smoothing length (denoted as radii), temperature (temp), density (rho), electron fraction (ye), and gravitational force (dgrav). The star is being disrupted by strong gravity when passing close to a Black Hole. When clusters are visually separated, they can be marked by drawing boundaries in the CoRPAS view using different colors, see Figure 5.3 (d). A linked view of the physical domain with highlighted groups of particles is shown in Figure 5.3 (e).

In the next example, we explore an SPH simulation of a binary system, see Figure 5.6. Two white dwarfs are bound together by gravity. The donor star gradually loses its mass, which flows to the heaviest white dwarf. The delivered material is heated when reaching the surface of the accreting star. The system consists of 40k particles with a
number of scalar fields, including radii and masses of particles, temperature, density, internal energy, and chemical components. For the projection in Figure 5.6 (a), we constructed a discrete plot (point size is 4 pixels) and several continuous representations weighted differently. They guided us when selecting five groups of particles as shown in Figure 5.6 (g). The corresponding configuration in physical domain is presented in Figure 5.6 (h): The detected groups of samples represent the disrupting star (green), the particles being transferred (blue), the heated particles that reached the surface of the accretor (red), the surface of the heaviest white dwarf (orange), and its core (yellow).

A clear advantage of continuous plots when compared to the discrete one is that the orientations of splats follow the paths of particles, i.e., they are aligned with an imaginary path over green-blue-red-orange groups. Thus, the CoRPAS reflect the underlying simulated process. In addition, it is now possible to see a border between the green and the orange groups, which is not visible on the discrete plot. The temperature-weighted plot helps to distinguish between the orange and the yellow clusters. Different weighting allows the user to draw conclusions about properties of particles in each group. The typically used volume weights cannot reveal all the groups because of the issue discussed in the previous section. Note that the relatively small blue group is dominant in Figure 5.6 (c), since the particles building the envelopes of the stars have large radii. The particles that reached the second star (red group) are heated and therefore have bright footprints in (f).

The use of logarithmically scaled plots is illustrated in Figure 5.7 for the same dataset. A continuous scatterplot (internal energy against particle smoothing length) is shown using linear and logarithmic scaling. The latter allows the user to identify several meaningful groups, whereas the former plot suffers from severe clutter. The four selected groups stand for: Donor-star (orange), accretor-star (green), accepted particles close to the surface of the accretor (blue), and particles in the envelope which may be lost by the system (magenta). Selections are done in the continuous scatterplot in Figure 5.7 (b) but shown in a discrete plot for the sake of exposition. We want to note that the blue and magenta groups in Figure 5.7 refer to the blue cluster in Figure 5.6 (g).
Figure 5.6: Projections of two white dwarfs dataset. Orientation of splats in CoRPAS reflect the actual evolution of particles on their way between two stars. It also allows to visually separate particles corresponding to the cores of different stars (green and orange groups in (g)). Different weighting helps to analyze the properties of particles in each group. Note that the volume-weighted CoRPAS (c) is not helpful for the analysis since it is strongly affected by outliers.
5. CONTINUOUS REPRESENTATION OF PROJECTED ATTRIBUTE SPACES

Figure 5.7: Logarithmic scaling of continuous scatterplot (smoothing length against internal energy, dataset as in Figure 5.6) allows for the detection of four groups, which is not possible in the linearly scaled equivalent.

Timings for computing CoRPAS of resolution $1024^2$ are presented in Table 5.1. All numerical images were performed on a PC with graphic card NVIDIA GTX 480 and implemented in CUDA. Doubling the parameter $k$ yields an increase of the splats’ area by a factor of four, thus, affecting the timing.

Table 5.1: Computation times for construction of CoRPAS.

<table>
<thead>
<tr>
<th>dataset</th>
<th>$k = 0.2$</th>
<th>$k = 0.4$</th>
<th>$k = 0.6$</th>
<th>$k = 0.8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>as in Figure 5.5 (right)</td>
<td>228.41ms</td>
<td>893.21ms</td>
<td>1953.41ms</td>
<td>3176.54ms</td>
</tr>
<tr>
<td>as in Figure 5.6 (d)</td>
<td>239.60ms</td>
<td>807.45ms</td>
<td>1588.36ms</td>
<td>2289.12ms</td>
</tr>
</tbody>
</table>
5.5 Conclusion

We presented an algorithm for CoRPAS, which is a part of a visualization system that links CoRPAS to visualizations in physical space. The user can intuitively and interactively change the projection matrix using star-coordinates widget, switch between linear and logarithmic plots, choose a proper weighting method and a spatial density kernel function, and mark regions of interest in the attribute space by drawing their boundaries with different colors.

The proposed method for continuous representation works on any spatial sampling including unstructured point-based data. The developed techniques are applied to various data, in particular, to SPH simulation data. We showed how continuous representations can help to better understand properties of given data, i.e., to find sharp boundaries and identify meaningful groups of samples. Finally, continuous scatterplots with logarithmic scaling were introduced and discussed. A texture mapping approach does not work in this case, since the shapes of the footprints depend non-linearly on their locations. However, a 1D look-up table can be effectively used instead, since the density $\sigma$ is radial.
Chapter 6

Multi-run similarity plots

This chapter is based on the publication:


Multi-run simulations are widely used to investigate how simulated processes evolve depending on varying initial conditions. Frequently, such simulations model the change of spatial phenomena over time. Isocontours have proven to be effective for the visual representation and analysis of 2D and 3D spatial scalar fields. We propose a novel visualization approach for multi-run simulation data based on isocontours. By introducing a distance function for isocontours, we generate a distance matrix used for a multidimensional scaling projection. Multiple simulation runs are represented by polylines in the projected view displaying change over time. We propose a fast calculation of isocontour differences based on a quasi-Monte Carlo approach. For interactive visual analysis, we support filtering and selection mechanisms on the multi-run plot and on linked views to physical space visualizations. Our approach can be effectively used for the visual representation of ensembles, for pattern and outlier detection, for the investigation of the influence of simulation parameters, and for a detailed analysis of the features detected. The proposed method is applicable to data of any spatial dimensionality and any spatial representation (gridded or unstructured). We validate our approach by performing a user
study on synthetic data and applying it to different types of multi-run spatio-temporal simulation data.

6.1 Introduction

Simulations of time-varying spatial phenomena are common in many fields of science and engineering. Typically, the simulations depend on initial conditions or input parameters. One of the research tasks is to understand the influence of the initial conditions on the result. Hence, the simulations are run multiple times with varying parameter settings. It, then, becomes a data analysis task to gain insight from the multi-run spatio-temporal data sets. The challenges are to efficiently handle the large amount of data and to effectively compare the outcome of multiple simulation runs, where the outcome comprises of time-varying 2D or 3D fields.

We propose a visual analysis approach to multi-run spatio-temporal simulation data that is based on a comparative overview visualization of the simulation runs. The two main questions are how to capture the similarity between simulation results and how to visually encode the multi-run similarities. Isocontours are a well-known instrument not only for scalar field visualization, but also for their quantitative and qualitative analyses [7, 9, 36–39, 82]. We propose an efficient computation of isocontour distances based on a quasi-Monte Carlo approach. Isocontour distances allow us to compare individual time steps of simulation runs. Hence, we can build a distance matrix, which we feed to a multidimensional scaling projection. In this projected view, simulation runs are visually encoded as polylines connecting consecutive time steps of a run. This view provides an intuitive overview of the multi-run data and supports further interactive visual analyses using selection and filtering operations as well as linked views to spatial data visualizations of the 2D or 3D physical space.

The projection-based multi-run data visualization approach is described in Section 6.2. Section 6.3 comprises computational aspects of the implementation including scalability issues. Parallel processing on both CPU and GPU is used where applicable. In Section 6.4, the interactive visual analysis approach with coordinated views is explained. Results are presented and discussed in Sections 6.5 and 6.6 for two types of
data, namely climate simulations (gridded 2D data fields) and astrophysical white dwarf simulations (unstructured point-based 3D data fields). Visualization of the parameter space is introduced in Section 6.7. In Sections 6.8 and 6.9 we consider synthetic data for the analysis of the proposed distance computation in comparison to the state of the art and for a user study on interactive visual data analysis, respectively. Finally, Section 6.10 consists of several subsections, where we discuss different topics of our approach, such as scalability, sampling, sensitivity and others.

The main contributions of our paper can be summarized as follows:

- A novel efficient computation of an isocontour distance metric is introduced. The isocontour distance computation is generally applicable to any (spatial) dimensionality and to any data structure (gridded or unstructured).

- A visual encoding of multi-run spatio-temporal data in form of a similarity plot of the runs’ individual time steps (short: multi-run plot) is proposed, which allows for the comprehension of the entire set of simulation runs. Interaction mechanisms allow for the analysis of the influence of initial conditions and for the detection of patterns and outliers.

- An interactive visual analysis system based on coordinated views of the multi-run plot and spatial data visualizations is proposed, where interactivity is achieved for large data sizes by moving the main computation step to a preprocessing step.

For the related work about multi-run spatio-temporal simulation data analysis we refer to Chapter 2.

We propose an approach, which tackles the aspect of multi-run spatio-temporal data visualization and analysis and allows for an exploration of the parameter space in conjunction with a physical space.
6.2 Multi-run Plot

The goal of this section is to construct a representative description for the time steps of all simulation runs in an ensemble, which can be used for an intuitive visual encoding of the ensemble by projecting the high-dimensional representative description to a lower-dimensional visual space. The idea of the proposed approach is to use time lines in a similarity plot, where the similarity is measured by looking at isocontours of individual time steps.

Since data are spatio-temporal, we investigate for each ensemble member a sequence of discrete time steps. The simulation state for every time step can be represented by a set of isocontours, or only by a single isocontour of a certain scalar field. Even when dealing with multi-field data, domain experts that run the simulations are frequently interested in analyzing a specific scalar field (which could also be a derived field like vector field divergence or curl magnitude). Considering isocontours of the selected scalar field, every ensemble member is represented by several threads (one per considered isovalue), where the threads represent the change of isocontours over time for the given time steps. Defining an appropriate isocontour distance function, we can use projection methods to generate a similarity (or distance) plot of all the samples to visualize the data. The points in the projection can be connected by polylines according to the threads they belong to.

6.2.1 Distance Function

Given a multi-run collection of time-varying scalar fields, our goal is to have a representative description of all runs rather than aggregate information over all runs such as mean or standard deviation. Isocontours have shown to be a suitable descriptor for scalar fields [7, 9, 36–39, 82] in addition to their property of being easily visualized and understood.

However, when projection methods are used to generate similarity or distance plots, some requirements have to be satisfied. First, a proper distance function has to be defined to generate a distance matrix. Second, since isocontours are defined implicitly, it
is important to describe all isocontours with the same accuracy for a fair comparison. Third, the isocontours do not only differ by enclosing areas, but have different shapes, which needs to be taken into account.

The concept of isosurface similarity maps was introduced by Bruckner and Möller [9]. They present structural information of a volume data set by depicting similarities between individual isosurfaces quantified by a robust information-theoretic measure. One obvious disadvantage of the approach is the high cost of generating the isosurface similarity map. The reported implementation can require several hours of processing time for a single data frame. Another approach for comparing 3D shapes by considering pockets in a complementary space is presented by Zhang [83], but it is also too slow for our purposes.

We propose to compute isocontour distances by comparing which areas in the global spatial domain are enclosed by which isocontours considering all time steps of all simulation runs. Hence, we need to integrate characteristic functions of the areas enclosed by isocontours. We propose to compute the integrals using a quasi-Monte Carlo (qMC) approach. The reasons for introducing this qMC approach are that it is faster than other existing approaches (such as the mutual information [9]), it can be used for spaces of any dimension, and it can be used for isocontours defined over any spatial data structure. The proposed isosurface similarity measure is described in details in Chapter 3.

6.2.2 Projection Algorithm

Having defined a proper distance function, it is possible to build a distance matrix (or dissimilarity matrix) $D = [d_{i,j}]$, $i,j = 1,\ldots,n$, where $n$ is the overall number of isocontours of all time steps of all simulation runs that are being analyzed. Our approach requires us to pairwise compare all $n$ isocontours, which is desirable, as we want to detect similarities of isocontours in the ensemble even if they occur at different points in time. Based on the distance matrix, we can apply a projection method to map the high-dimensional binary vectors to a visual space for the multi-run plot.

Many projection methods exist that target different objective functions. Multidimensional scaling methods such as MDS [84], IsoMap [85], or Locally Linear Em-
bedding [86] are widely used for the visual representation of high-dimensional data. Considering the large data sizes and large matrices we have to process, we aimed for simplicity and least computational complexity. Therefore, we decided to choose the MDS approach by Wickelmaier [87].

Our algorithm can be summarized as follows:

1. Set up matrix of squared distances $P = [d_{i,j}^2]$.
2. Apply double centering: $B = -\frac{1}{2} \cdot J \cdot P \cdot J$, where $J = I - \frac{1}{n} \cdot E$, $I$ is the identity matrix, $E$ is the matrix with all entries being 1, and $n$ is the number of samples.
3. Extract the $m$ largest positive eigenvalues $\lambda_1, \ldots, \lambda_m$ of $B$ and the corresponding $m$ eigenvectors $v_1, \ldots, v_m$.
4. An $m$-dimensional spatial configuration of the $n$ objects is derived from the coordinate matrix $X = V_m \cdot \Lambda_m^{1/2}$, where $V_m$ is the matrix of the $m$ eigenvectors and $\Lambda_m$ is the diagonal matrix of the $m$ eigenvalues of $B$.

Here, $m$ is the dimensionality of the space into which we project, i.e., 2 or 3 for 2D or 3D visual spaces, respectively. If the intrinsic dimensionality of the data is larger than three, different combinations of the respective eigenvectors can be chosen interactively for further investigation.

### 6.3 Scalability of Implementation

The goal of our approach is to allow for the interactive visual analysis of multi-run data. Despite the large size of multi-run data sets, we aimed for interactivity and a smooth user experience when running the algorithms on a commodity off-the-shelf PC workstation or even a laptop. Thus, run-time computation costs should be sufficiently low, but also pre-computation times should remain as small as possible. In this section, we want to discuss the aspect of scalability of our implementation. We can subdivide the data handling pipeline into four stages as depicted in Figure 6.1, which allows for an efficient usage of computational resources and a limited amount of disk access. In the following, we will discuss each of the four stages.
Figure 6.1: The proposed four-stage scheme of the data handling process.

Loading of data: The first issue that needs to be addressed is the loading of data to main memory. Even for simulations with modest spatial resolutions, the amount of time steps computed make a single run easily exceed 1 GB. Thus, basically all real-world multi-run spatio-temporal data sets do not fit into the main memory of today’s PCs. Our approach only needs access to the data field when computing the descriptive vector of an isocontour. This computation is restricted to a single time step of a single run. Hence, we never need to load multiple time steps and/or multiple runs into main memory simultaneously.

Additionally, we can speed up data loading by skipping those time steps, where the chosen isovalues do not lie within the respective scalar field’s range. Hence, as an optional pre-processing step, we can compute the range of all scalar fields. Then, we can check against those intervals during loading.

Preparation of isocontours: Storing all isocontours that are relevant for the multi-run analysis would significantly exceed the memory constraints. Fortunately, we do not need to extract the isocontours. For the generation of the descriptive vector, we test against the implicit isocontour representation. The computational step reduces to interpolating the scalar field’s value at the random points and testing against the isovalue.

For gridded data, no further computations are necessary. For unstructured data, one may have to apply some affine transformations to make individual scalar fields comparable to each other, see Section 6.6. Computing the affine transformation requires a one-time access to each scalar field.

Distance matrix calculation: For the distance matrix calculation, we first need to generate the set of random points, which is done once for the entire multi-run data set. Then, the descriptive vectors of the isocontours are computed, which requires the data

Projection calculation
loading and isocontour preparation steps mentioned above. The respective interpolation step can be easily parallelized and executed on a GPU using a CUDA implementation. Finally, the distances are computed and inserted into the distance matrix. Since the matrix is symmetric and its diagonal entries are zero, only \( \frac{n^2}{2} - n \) distances need to be calculated. However, the matrix can be rather large and the descriptive vectors may be large, as well. Thus, this step may require a considerable amount of computations, but it can also be sped up using parallel computations on the GPU in a straight-forward manner.

**Projection calculation:** All the steps described above can be pre-computed. During the interactive analysis step, we only need the distance matrix, from which we compute the projection using all entries or a selected subset. Actually, the projection for all entries can also be pre-computed. Selected subsets are typically small, such that a recomputation of the projection is fast enough for an interactive setting when using a parallel implementation on the GPU.

The size of the distance matrix may impose a problem though. For example, when considering four isocontours for four simulation runs with 1,024 time steps, the distance matrix is of size \( 2^{14} \times 2^{14} \). Assuming single precision, this amounts to 1 GB of memory. Since the projection computation involves matrix multiplications with temporary variables etc., the required memory size is about four times the size of the matrix, i.e., about 4 GB. Moreover, the memory requirements increase quadratically with the number of considered isocontours. For such large matrices even a simple computation of a projection may require hours.

Fortunately, we can exploit temporal coherence of the simulation runs. Only considering every \( n \)-th time step for all simulation runs does not affect the projection result much for small \( n \). Thus, we can significantly downsample in the temporal dimension to reduce the size of the distance matrix.

**Parallelization:** For efficient data processing, we implement a pipeline of steps that are executed in parallel using multiple threads. The main pipeline steps are loading data from hard disk, preparing the data for interpolation, calculating states of the random points, and releasing the data. Some of these steps are further subdivided into substeps and others rely on a CUDA implementation (e.g., for scattered data interpolation).
6.4 Visual Analysis Based on Multi-run Plot

The multi-run plot serves as an intuitive overview of the ensemble data set. Still, when selecting multiple isovalues and depicting the threads for all runs, the plot may get cluttered. Hence, we provide interactive filtering and selection methods for the plot.

After selecting the scalar field to be investigated, a typical interactive multi-run data analysis involves several steps: (1) Choose representative isovalues; (2) examine temporal changes and identify patterns and outliers; (3) examine linked views to spatial visualizations; (4) select subsets for detailed investigation in a reconfigured view; and (5) analyze the influence of input parameters or initial settings. Of course, the five steps do not need to be strictly executed in the given order and one may iterate through the steps within an interactive analysis.

To explain and demonstrate those interactive analysis steps, we use a simple synthetic dataset that we created and show how the features in the data can be revealed. We generated three simulation runs over a 2D grid of size $512 \times 512$ consisting of 100 time steps of a single scalar field. The scalar field represents four peaks of Gaussian kernels (with absolute values from 0.0 to 0.25). Their values accumulate when they overlay. Initially, the four peaks coincide at the center of the domain. Then, they start to rotate around the center and diverge from the center. The process is schematically shown in Figure 6.2: One simulation run (green arrows) immediately separates the four peaks, while the other two runs (red and blue arrows) separate only two pairs of peaks. Afterwards, the pairs of peaks of one of the latter runs (blue arrows) also split, thus creating four separate peaks similar to the first run (green arrows). Our goal is to identify the behavioral patterns and their features.

The quality of the multi-run plot obviously depends on the choice of suitable isovalues (Step 1). A badly chosen isovalue can lead to a low-quality data representation. Typically, a domain expert who performs the analysis of his/her data knows about the underlying simulation and his/her expertise allows for a suitable manual selection of relevant isovalues. Otherwise, it is possible to sample the scalar field’s range with a larger number of isovalues, e.g., by using equidistant sampling in a linear or logarithmic scale. A typical scenario for an interactive analysis of a multi-run data set would then start...
with an overview over all selected isocontours for a single or a few runs. A transfer function shall be applied to the range of the scalar field to color-code the plot according to the displayed isovalues. This first view on the multi-run data allows for a fast identification of possibly interesting isovalues. Those can be interactively selected for the analysis of all simulation runs. Alternatively, the choice of a representative isovalue can be made according to other visual analysis tools for scalar fields like the isosurface similarity map [9], cf. Section 6.8, where we use it for the evaluation of our distance function.

For the synthetic example, we know that the range of the field is initially $[0, 1]$ and decreases to $[0, 0.25]$ when the four peaks split. Hence, we know that any isovalue out of the range $[0, 0.25]$ would be suitable and pick isovalue 0.01. If we had not known much about the simulation, we would have started by sampling the range and examining the respective similarity plot of the respective isocontours. Figure 6.3 shows the similarity plots for ten isocontours for (a) the “red” and (b) the “green” simulation run. For the computation of the distances between isocontours we use 32,768 random points. Both plots show that the polylines for the different isovalue have a similar shape. Hence, choosing the single isovalue 0.01 sufficiently captures the scalar field.
6. MULTI-RUN SIMILARITY PLOTS

![Similarity plot for 10 isocontours (red run)](image)

![Similarity plot for 10 isocontours (green run)](image)

Figure 6.3: Similarity plots of isocontours (changing over time) of synthetic data for “red” (a) and “green” (b) runs (see Figure 6.2). The ten threads correspond to ten isovalues chosen equidistantly from $[0, 0.25]$. Color encodes isovalue using a rainbow color map.

Having chosen suitable isovalues, one generates the multi-run plot to investigate changes over time and to detect patterns and outliers in the entire ensemble (Step 2). Figure 6.4 (a) shows the multi-run plot for the three runs of our synthetic example for isovalue 0.01. Color encodes the three runs according to the arrows in Figure 6.2. Note that points that are located close to each other in the plot correspond to isocontours with similar shapes, independent of the time step they belong to. In particular, as the polylines connect points in temporal order, self-intersection of the polylines may occur.

![Multi-run plot](image)

![Time step 50](image)

![Time step 71](image)

![Time step 100](image)

Figure 6.4: (a) Multi-run plot using a 2D MDS projection of the synthetic data. (b-d) Linked spatial views of isocontours for different time steps. Color encodes three different runs.
6.4. VISUAL ANALYSIS BASED ON MULTI-RUN PLOT

When examining the plot, one can observe that all polylines start in the same location (to the left) indicating the same initial state. Then, the green polyline separates immediately, while the red and blue ones coincide for quite some time (having the same evolution) until they start separating, as well. Finally, the green and blue polylines merge again and end up in the same location, while the red polyline ends at a different location. Moreover, we observe oscillations in form of a pendular pattern in the polylines, which correspond to the rotational behavior around the center of the domain. We can conclude that the multi-run plot exhibits all features and patterns in the dataset according to the modeled behavior (see Figure 6.2).

Having identified relevant structures, it is of interest to explore shapes, areas, and location of the respective isocontours. We support such an analysis by providing linked views between the multi-run plot and spatial data visualizations using isocontour rendering (Step 3). When selecting a single point in the multi-run plot, the respective isocontour is displayed in the linked view. One may also select multiple points in the multi-run plot and compare the isocontours next to each other in multiple linked views or embedded into one linked view using overlays. Another option is to animate the isocontours over time in the spatial view and highlight the currently displayed point(s) in the multi-run plot.

Figure 6.4 (b-d) shows three selected frames of the animation, while Figure 6.4 (a) shows the respective points in the multi-run plot. It can be observed that the blue and the red run are indeed still identical at time step 50 (b) and that the blue and the green
run are identical at time step 100 (d), while time step 71 is part of the transition (c).

Our interaction mechanisms also allow for drilling down on the multi-run plot (Step 4). One can select a region of interest in the plot and reconfigure the plot to only display the selected points, i.e., we recompute the projection for the selection. The recomputation is fast and allows for interactive updates, as the already computed distance matrix can be re-used. Note that the reconfigured plot is not affected by points that were not selected. Figure 6.5 (b) shows a reconfiguration of the multi-run plot for our synthetic dataset according to the selection in Figure 6.5 (a). The reconfigured plot allows us to observe that the green run also exhibits a pendular pattern (due to the modeled rotation around the center).

Another reconfiguration option for the multi-run plot is that we also support interactive switching between projections to 2D and 3D visual spaces. Although 3D plots are perceptually more challenging and require interactive rotations to fully understand them, we observed that the additional dimension can be helpful. For example, in Figure 6.4 (a) we observed a pendular behavior in the 2D plot, while the 3D plot in Figure 6.6 clearly shows the rotational behavior. Also, the 3D plot documents that the green run only coincides with points of the other runs in the initial and final state, but not in between, which was not clear in the 2D plot. The benefit of the third dimension, of course, depends on the data, see Sections 6.10.3 and 6.10.4 for a discussion.

![Multi-run plot using 3D MDS projection on synthetic dataset.](image)

(a) Orthogonal view  
(b) Side view

Figure 6.6: Multi-run plot using 3D MDS projection on synthetic dataset. The 3D visual space is shown using (a) a view orthogonal to the 2D MDS projection in Figure 6.4 (a) and (b) a side view.
Finally, our interactive analysis tool also supports the investigation of the influence of the input parameters (Step 5). To do so, we apply a transfer function to the range of values for the tested input parameters and color code the runs by using the respective colors for the polylines.

Overall, there have been multiple ways we apply transfer functions (or color maps) to the multi-run plot to investigate certain aspects. We have seen that we can color-code each polyline according to the isovalue it represents or according to which simulation run it belongs to. One could also encode the temporal dimension, i.e., there would be a color change within the polyline. The applied transfer functions can also be coupled by using not-interfering color maps. For example, one would change the hue to distinguish the different runs, while brightness is used to encode the time steps. This has been done in Figure 6.6 such that it is clear which endpoint of the polyline represents the initial state (darker) and which the final state (brighter). For the choice of transfer functions, one needs to consider that they typically encode numerical attributes (isovalue, time step, or input parameter) such that a continuously changing transfer function shall be applied. In case the multiple runs do not depend on a numerical input parameter but are an unsorted collection of runs (such as for the synthetic dataset), belonging to a run is a categorical attribute and shall be encoded by clearly distinguishable colors (e.g., red, green, and blue for the given example).

6.5 Application to Gridded Data Analysis

Now, we want to apply our techniques to real datasets. The gridded data we are using represent a multi-run simulation of a 2D climate model. Initially, a single run over about 200 years is simulated. Then, random time points of that run are used as initial conditions for multi-run simulation over one to three years. The multi-run datasets consist of eight to eleven simulation runs with 1,460 time steps per year. Each simulation time step consists of multiple scalar fields such as sea ice thickness or sea surface temperature. We use the output ported to a regular 2D spatial grid of size $192 \times 96$. Hence, we can use bilinear interpolation for the generation of the distance matrix. For its computation we evaluate the scalar fields at 32,768 random points. We exploit temporal coherence
6. MULTI-RUN SIMILARITY PLOTS

Figure 6.7: (a) Multi-run plot of gridded climate data for different scalar fields. Colors indicate the five isovalues. Threads have pendular or loop behavior representing annual patterns. (b) Recomputed projection for the points of the “blue” group (sea ice thickness = 0.95m) from (a) after selecting the third year only. Colors indicate different runs.

to keep the matrix size small enough to fit hardware and system requirements by only considering every seventh time step of all the runs.

Figure 6.7 (a) shows the plot for three simulation runs of 3 years for sea ice thickness. The threads are color-coded according to the five selected isovalues. It can be observed that the $3 \times 5 = 15$ threads are grouped by isovalue and that all polylines exhibit the expected repetitive patterns that represent annual oscillations. Interestingly, some polylines describe a pendular pattern, while others describe a loop pattern. The most diverse threads seem to exist for the iso value for sea ice thickness shown in blue color (sea ice thickness $= 0.95m$). Figure 6.7 (b) shows a further analysis of this iso value. Now, the plot shows these simulation runs for the last year only. The colors have been changed to encode the different runs.

In order to analyze, in which spatial regions the runs differ, we apply the linked view to spatial visualizations. The linked view shows an animation over time of the isocontours for the three runs in an overlaid mode. Figure 6.8 shows one frame of the animation for the time points highlighted in Figure 6.7 (b). It can be observed that there is a clear difference in ice distribution for the three runs. Furthermore, the similarity plot in Figure 6.7 (b) suggests that the green and blue simulations are more similar than the red one, which can be confirmed with the linked spatial view. In particular, the red simulation is the only one, where the north-east corner of the map remains fully covered.
The analyses have been carried out together with domain experts and the benefits of our techniques have been discussed. The domain experts pointed out the following aspects:

- The multi-run plots allow for an efficient and effective analysis of the ensemble to detect patterns and find outliers. In particular, comparisons of runs are easy to perform.

- The provided linked view to spatial visualization allows for an interactive assessment of the spatial distribution and allows for an immediate identification of spatial regions of interest. This substantially saves time by eliminating the need for using separate visualization packages and manually combining them with the analysis process by calling the procedures for potentially interesting regions of interest.

- A potential extension of the tool may be to add automatic components for the detection of spatial regions of high relative or absolute changes.
6.6 Application to Unstructured Data Analysis

The unstructured data we are using represent a 3D astrophysical multi-run smoothed particle hydrodynamics (SPH) simulation of two merging stars. The matter of the smaller star flows to the larger star over time. The multi-run simulation has two main initial parameters, which are the values for the masses of the two stars. The multi-run data set consists of 45 runs with 700 to 1,500 time steps. Each simulation run uses 39,718 particles that change their positions over time and carry information about mass, internal energy, density, temperature, etc. We consider internal energy for our analysis. For the computation of the distance matrix, we use the SPH interpolation kernel. Since this interpolation scheme is significantly more expensive than the bi- or trilinear interpolation schemes we use for gridded data, computation times are larger for SPH data. In our tests, we observed that using 8,192 random points are enough for producing a high-quality projection and still allows for a very fast processing, see Section 6.8 for further discussion. Also we exploit temporal coherence by only considering every second time point of the simulation runs. As temporal down-sampling may lead to visual artifacts in form of less smooth polyline representations, we replace the polylines with third-order interpolating Catmull-Rom spline curves.

Before we define the positions of our random points, we have to identify their domain. Since particle simulations do not have spatial boundaries, we have to define a global domain. The global domain is determined as the smallest bounding sphere that encapsulates all particle positions of all simulations at all points in time.

For the isosurface extraction from particle data, we use the approach presented by Rosenthal and Linsen [88], which is fast, as they directly extract isosurfaces in point-cloud representation using linear interpolation. We decided to plot the point cloud instead of using more advanced point-based rendering approaches (such as splat-based renderings as shown in Figure 6.10 (b), cf. [48]) for multiple reasons: renderings are faster, allow for partially seeing multiple nested surfaces, and have less problems with noisy isosurfaces.

Figure 6.9 (a) shows the multi-run plot for a selected isovalue and all simulation runs using 266 time steps from desired time intervals of each simulation run. Color is
used to encode the different runs. The projection exhibits a clear pattern with three main parts (top, middle, and bottom). However, when observing a large ensemble with many runs, it may be that the main pattern occludes more subtle patterns or outliers. Thus, we propose an alternative rendering of the plot by first aggregating the line density per pixel and then color coding the density according to a suitable color map. Figure 6.9 (b) shows the respective density representation and the applied color map. The plot also exhibits the three parts of the main pattern, but also reveals an outlier in the top-middle area that was hidden in Figure 6.9 (a). The aggregated plot helps the user to get an overview, while the non-aggregated plot is needed for comparative analyses of individual plots. Our tool allows for interactive switching between the two rendering modes.

Figure 6.10 (a) shows a 3D view of a single run of the plot in Figure 6.9 (a). The 3D plot reveals that the top part of the pattern forms a ring, the transition part in the middle is a spiral, and the bottom part forms a cluster. The same holds true for all other runs, but there are some differences. For an investigation on how the input parameter influences the simulation results, one can color-code the threads according to the parameter’s value. Here we encode as parameter value the starting mass of one star. Figures 6.11 (a) and (b) show two orthogonal views on the multi-run plots, where the yellow color is brighter when the mass is lower. We generate the views by using the first- and second-largest eigenvalues (a) and second- and third-largest eigenvalues (b) for projection. All threads lie within the shape of a cone. All threads start on the base of the cone and, after the
transition phase, end up in the cluster at the tip of the cone. The runs with higher masses (darker) spiral on the outside of the cone from the base towards the tip, while the runs with lighter masses (brighter) move early on to the inside of the cone. We observe a continuous transition between the behaviors. To further investigate what is happening along the threads, we selected four keyframes in the plot and investigate the physical space in linked views, see Figure 6.11 (c)-(f). Figure 6.11 (c) shows that the base of the cone represents the initial stage, where the two stars are fully separated. Then, the stars with higher masses merge more quickly (d), while for the stars with lower masses the merging phase takes longer (e). Eventually, all threads end up in a fully merged system (f) at the tip of the cone.

The multi-run plots above suggest that there is some rotational pattern. Indeed, the system rotates during the entire simulation, which can be quickly confirmed using linked view to the physical space. Since the rotational pattern is rather dominant, one may consider in an additional analysis step to exclude the rotational movement in a pre-processing step. This can be achieved by aligning the orientation of isosurfaces at different time steps using principal component analysis. Figure 6.12 (b) shows the multi-run plot from above after rotational alignment. We can observe that the orbiting behavior has been removed and the three stages of the simulation become even more obvious. In addition, the rotational pattern does not occlude other structures anymore and it is easier to compare the different simulation runs. In particular, one can see
the smooth transition from bright to dark threads. Interestingly, we can obtain a qualitatively similar plot without applying rotational alignment, namely, by considering further eigenvalues for the projection. Using the first- and sixth-largest eigenvalue, we obtain
Figure 6.12: (a) Using first- and sixth-largest eigenvalues for projections in multi-run plot (as in Fig. 6.11). (b) Using first- and second-largest eigenvalues for projections in multi-run plot after rotational alignment. Both views do not exhibit the main rotational pattern and are qualitatively similar.

the multi-run plot in Figure 6.12 (a), which is an orthogonal view to the ones in Figure 6.11. Note than we can also detect the outlier (mid right in Figure 6.12 (a) and mid left in Figure 6.12 (b)), which we already had identified in the density representation of Figure 6.9 (b). Such a result confirms that our approach is able to capture the data features and reflect them in the projection space. The whole set of 2D projections up to the instrinsic dimensionality of the data can be investigated visually by looking at a corresponding scatterplot matrices, which show the projection views for all the pairs of extracted dimensions.

Figure 6.13 shows a few steps of further interactive analysis using linked views. Figure 6.13 (d) shows a filtered view of the projection shown in Figure 6.12 (b) by selecting a single run. The plot conveys again that there is a initial phase with many time steps, a changing phase with few time steps, and an final phase with again many time steps. The user selects a keyframe from each phase for one thread and investigates the respective isosurface in the linked views shown in Figure 6.13 (a)-(c). The identified phases correspond to the main phases of the astrophysical simulation.

Also for this application, we conducted the analysis together with a domain expert to analyze all features. For example, we observed oscillations at the beginning phase of the simulations and the domain expert was able to recognize them as mass transfer...
oscillations. Hence, we can conclude that our approach is sensitive enough to even capture local data features. We also asked the domain expert to judge our techniques and he pointed out the following:

- Using the multi-run plots, the expert was able to recognize all simulation features, in particular, the three main stages (initial, merging, and final), the influence of the initial parameters, and the mass transfer oscillations during the initial phase. In particular, our multi-run plots incorporated all features that are known to the expert and did not miss any.

- The discovered features were known to the expert, as he has been analyzing this dataset for years, but he would have been able to detect them much more efficiently using our techniques. In particular, the multi-run plots allow for an immediate detection of the transitions between the three phases. In combination with the linked views they allow for the selection and investigation of a simulation state during the merging phase immediately for all runs, instead of calculating
orbital separation, seeking the merging moment, and using third-party packages
to visualize each selected time step.

- A potential extension of our tool would be to compute further valuable application-
specific information (like orbital separation) to guide the user directly to the im-
portant features.

6.7 Parameter Space Visualization

Beside of visual comparison of the multi-run simulations, identification of their features
and behavior patterns, analysis of the parameters’ space is of interest. We propose to
color code simulations on the similarity plot according to their parameters’ values using
one- and two-dimensional transfer functions, depending on purposes of the analysis. For
the visualization of the simulations in the parameter space, we propose to build a scatter-
plot, where positions of the splats reflect the parameters’ values of the simulations,
and a color of the splat is taken according to the chosen color scheme.

We consider two types of simulations’ parameters: categorical and numerical. Com-
plex initial conditions (as for the climate simulation data in our examples) we can also
consider as a categorical parameter. In this case we propose to assign to different sim-
ulations colors with a highest possible difference in hue, because such a parameter can
not be ordered and the only purpose is to distinguish different simulations visually, see
Figure 6.7.

In the case of single numerical parameter we propose to use one-dimensional trans-
fer function to describe the parameter’s range. An interactively adjustable transfer func-
tion allows to highlight the simulations within a certain parameter range out of the whole
range. Interactive linking between the projection space and parameter space allows to
understand a structure of the projection, detect dependencies and identify outliers.

For ensembles with more than one simulation’s parameter, we propose to use pair-
wise visualization of the parameter space. The restriction for the number of simultane-
ously considered parameters is natural and based on a number of independent compo-
nents of the colors and perception of similar colors. There are different ways to generate
6.7. PARAMETER SPACE VISUALIZATION

Figure 6.14: On (a) a 2D scatterplot of the SPH simulation data visualized in the parameter space is shown. An RGB interpolation scheme is used for the two-dimensional transfer function. The 3D MDS projections with a side and top views are shown on (b) and (c) correspondingly. The main principal component (vertical axis) of the MDS projection for the SPH data set plotted over time (horizontal axis) is shown on (d). The colors correspond to the color scheme on (a).

A two-dimensional color scheme. A common way to do so is to use two one-dimensional transfer functions and combine them using different combining operators. In addition, we also support the generation of two-dimensional transfer functions by choosing colors for the corners of the given parameter space range (which forms a rectangle in case of two parameters selected) and interpolating the colors using bilinear interpolation (other interpolations schemes are also applicable, such as barycentric, or scattered data interpolation).
We show such application of the two-dimensional transfer function on the astrophysical data set, see Figure 6.14. The SPH data set we are using contains information about two parameters, which are masses of the two stars. The stars are equivalent (pairs \((a, b)\) and \((b, a)\) are equal), therefore, a scatterplot visualization of the parameter space has a triangular shape. In this particular case a combination of two one-dimensional transfer functions is not the best solution, because a half of the parameter space is not represented. In contrast, a two-dimensional color scheme allows to highlight the main data features, namely, a difference between the star masses and their total mass. These features are represented by the directions along the biggest edge of the triangle and its distance to this edge. In Figure 6.14 (a) we show such visualization of the parameter space using three different colors for the angles of the triangular scatterplot. The blue color corresponds to the simulations with low total masses, red color to the simulations with high total masses, green color to the simulations with the high differences between the star masses. In this example we use interpolation between RGB components of each color. The color scheme allows to clearly distinguish dependencies between the simulations’ colors and locations on the projections.

Having simulation parameters and a color scheme selected, we propose to use linked views of parameter space and the multi-run plots. Using a two-dimensional transfer function defined over the parameter space, we map the respective color to the multi-run plots. For further interactions, we implemented selections and corresponding highlighting in both directions: parameters $\leftrightarrow$ simulation runs. Such interaction allows to select individual simulation runs or group of them to see their parameters, as well as filter simulations by selecting certain parameters in order to investigate possible dependencies.

From the projections shown in Figure 6.14 (b, c) we can conclude that the outer part of the cone’s base consists of simulations with low total masses. Simulations with higher total masses are located below, but still on the outer side of the cone. Note that all the simulations with a high difference between the star masses are located inside of the cone. All the dependencies show a smooth color and location transitions. However, there is one outlier that can be spotted out in the center of the Figure 6.14 (c), which can be easily linked to the parameter space and identified as a simulation with the highest difference between the star masses. This can be confirmed by looking at the visualization of the main principal component of the MDS projection over time, see
Figure 6.15: On (a) a 2D scatterplot of the SPH simulation data visualized in the parameter space is shown. An HSV interpolation scheme is used for the two-dimensional transfer function. The main principal component (vertical axis) of the MDS projection for the SPH data set plotted over time (horizontal axis) is shown on (b). The colors correspond to the color scheme on (a).

Figure 6.14 (d). Note that this is the outlier we have detected before, see Figure 6.9 (b). Using the parameter space visualization, we see that the configuration of the two stars with the biggest difference, namely, configuration of stars with 0.65 and 1.05 solar masses, led to the outlier.

To highlight this dependency we can modify the color scheme shown in Figure 6.14 (a) by using the same color for the simulations with the same star masses. Hence, we use a one-dimensional transfer function, but we do not apply it to one of the input parameters, but to a combination of them. For a better perception of the colors, we also apply interpolation between HSV color components instead of RGB. The resulting visualization of the parameter space is shown in Figure 6.15 (a). Now it is even clearer, that results of the simulations, i.e., their final states, are more deviating, as larger the differences between the star masses, see Figure 6.15 (b).

We can conclude that an interaction with a visual representation of a parameter space and proper color schemes allow for a fruitful analysis of the multi-run simulation results. Such a visualization helps to understand a structure of the projection and to highlight different groups of simulations. Doing so, one can narrow down the analysis area and perform detailed analysis of the possible dependencies and outliers.
6.8 Distance Function Analysis

In order to estimate the usefulness of the introduced distance function we compared it with the state of the art, namely the information-theoretic measure of similarity based on mutual information [9]. For the comparison we used the synthetic data from Section 6.4 shown in Figure 6.4.

The outcome of the distance computations can be displayed using similarity maps that plot the distance matrix with grayscale values ranging from black (distance 0) to white (distance 1). We consider isocontours for isovalue 0.01 and produce similarity maps for all time steps of the three runs. Figure 6.16 (a) depicts the similarity map when using mutual information, while Figure 6.16 (b) depicts the similarity map when using our distance computation. Since the isocontours in the three runs change smoothly and gradually over time, one would expect to see nine blocks of smooth diagonal structures. Indeed, the similarity map produced with our distance computation exhibits the expected structure. The mutual information approach, however, clearly exhibits some artifacts that cannot be explained from the simulation aspect. It is worth mentioning that the mutual information similarity was measured with the highest resolution ($512 \times 512 = 262,144$ pixels) of the corresponding distance transform and joint histogram, while only $32,768$ random points were used for our approach.

We also investigated the impact of the distance computation result on the multi-run plot. Figure 6.16 (c) shows the 2D multi-run plot corresponding to the distance matrix represented in Figure 6.16 (a). The multi-run plot corresponding to our result is shown in Figure 6.4 (a). It can be observed that the multi-run plot is negatively affected by the artifacts in the mutual information similarity map. The patterns of the runs cannot be recognized well due to a cluttered self-intersecting structure. When using a 3D MDS projection and selecting an orthogonal view, see Figure 6.16 (d) to the view in Figure 6.16 (c), one can actually observe the rotational pattern of the "red" and "blue" runs, but their circular structures are distorted (as opposed to the clear circular structure in Figure 6.6).

As already mentioned by Bruckner and Möller [9], the main disadvantage of using mutual information for isosurface distance computation is the considerably high compu-
6.8. DISTANCE FUNCTION ANALYSIS

(a) Mutual information
(b) qMC approach
(c) Multi-run plot
(d) Orthogonal view

Figure 6.16: Similarity maps of synthetic multi-run data calculated by mutual information (a) and our dissimilarity measure (b). 2D multi-run plot (c) corresponding to similarity map (a) and its orthogonal view (d) in a 3D multi-run plot.

tional costs. For fair comparison we implemented both approaches without using GPU implementations. The times spent for the generation of the distance matrices shown in Figure 6.16 (a, b) are 628.5s for the mutual information approach and 14.5s (both times without data loading) for our approach, i.e., our approach was 43 times faster. Moreover, when calculating mutual information for all time steps of all runs, the information about the corresponding distance transforms has to be stored in system memory to avoid recomputing them for each comparison. Hence, to store one distance transform using single precision for the full synthetic data resolution, $512 \times 512 \times 4B = 1MB$ of memory is required, while our approach only requires us to store 32,768 Boolean values for each isocontour using only $4KB$.

To handle larger data sets, Bruckner and Möller [9] suggested to downsample the resolution of the distance transform. They reported acceptable results when downsam-
Figure 6.17: Similarity maps (a,b) of synthetic multi-run data calculated by different dissimilarity measures using downsampled resolution. Corresponding 2D multi-run plots (c,d).

We downsampling each dimension 8 times. We created the distance matrices and the multi-run plots for the synthetic data using decreased resolution for both approaches. For our approach we took 4,096 instead of 32,768 random points and for the mutual information calculations we used $64 \times 64 = 4,096$ instead of $512 \times 512 = 262,144$ pixels. The results are shown in Figure 6.17. We observe that the downsampling did not affect much the result of our approach, while in the mutual information approach the artifacts were enhanced. It can be confirmed by comparing the corresponding multi-run plots. Figure 6.17 (c) is severely different than Figure 6.16 (c), while Figure 6.17 (d) is very similar to Figure 6.4 (a). Finally, the times of the calculation of distances without data loading are 28s for the mutual information and 2s for our approach.
The benefits of our quasi-Monte Carlo approach compared to the mutual information approach for isocontour distance calculation can be summarized as follows:

- Significantly better performance (one order of magnitude).
- Significantly less memory consumption (two orders of magnitude).
- Precise distance matrix calculation for a stable MDS projection output.
- Applicable to any kind of data, including unstructured data types.

6.9 User Study of Visual Analysis

Since our goal was to develop an approach for the visual analysis of multi-run simulations independently of the underlying field of science, it is important to validate that users with different background can interpret our visual representation and operate with our tool to interactively analyze data. Therefore, we performed a user study with 18 participants from different fields of science and with different scientific grades. Among the subjects were six post-doctoral fellows, nine PhD students, and two undergraduate students from different fields of science including computer science, mathematics, chemistry, biology, physics, oceanography, and geoscience. None of the subjects was familiar with our tools nor with the tasks of multi-run data analysis.

The user study concentrated on the main components of the visual analysis of multi-run simulations. First, the subjects were supposed to detect the simulation features and identify behavioral patterns. Then, they had to compare a number of simulation runs to draw conclusions about initial parameter influence and to estimate corresponding dependencies. Finally, they were asked to search for possible outliers.

We created three synthetic multi-run ensembles based on the synthetic data from Section 6.4. More precisely, we took the “red” simulation and produced variants thereof by introducing three different parameters, which were designed to affect the rotational velocity of the spots around the center of the domain. For each of the three generated ensembles only one parameter changed its values and had to be investigated. The simulation parameters were size of the spots, shape of the spots, and distance between the
spots. Each of the three ensembles consisted of nine simulation runs and the parameters
changed gradually with a constant step size. We manually modified the ensembles by
introducing for each of the three ensembles an outlier.

The subjects were asked to answer three questions for each data set: (1) What is
the simulation parameter? (2) How does the simulation parameter affect the rotational
velocity? (3) Is there an outlier in the ensemble (if yes, identify it)? For the analysis, the
subjects used two tools: (A) our visual analysis tool based on multi-run plots, respec-
tive interaction methods, and coordinated views as described above, and (B) a tool that
allowed the subjects to visually inspect the isocontours used in our tool individually or
in an animation over time. The second tool is what application scientists typically use,
namely, visualization methods that allow for the visualization of an individual time step
of an individual run or respectively generated animations (often only movies) that show
the change over time.

Before the subjects were confronted with the data sets to be analyzed, there was a
5-7 minutes training, where the subjects were explained what multi-run simulations are,
what the purpose of the study was, and how to use the two tools. We randomly picked
one of the three ensemble data sets (for both tools) for training the subjects. Then, one
of the remaining ensembles was randomly picked for tool A and the last remaining en-
semble for tool B. By using different data sets for different tools we avoided a learning
effect. We also randomized the order, i.e., whether the subjects start with tool A or tool
B. During the study, we recorded timings for completion and asked the subjects how in-
tuitive the tool was for the given analysis task on a five-step Likert scale (1 being worst,
5 being best). After the study, we asked the subjects about the general intuitiveness and
usefulness of the tools, requested further comments, and computed the correctness of
the given answers.

The results of our statistical analysis of the user study are given in Table 6.1 and in
Figure 6.18. It can be observed that our analysis tool A led to faster answers, in general,
and to more accurate results for Tasks 2 and 3. Also, the subjects rated that our tool
was easier to use for each of the three tasks. In general, they rated that our tool was not
less intuitive than tool B and was considered more useful. All these statements were
checked for statistical significance by Welch’s $t$-test with $p = 0.05$ and $df = 32$. 
6.9. USER STUDY OF VISUAL ANALYSIS

Figure 6.18: Box plot of user evaluation. Yellow and blue colors indicate tools A and B, respectively. Green font indicates comparisons with statistically significant differences, otherwise red font is used.

Table 6.1: Statistical analysis of user study. Our tool A, generally, led to faster and more accurate analyses and was judged to be easier and more useful.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Average study time</th>
<th>Standart deviation</th>
<th>Statistically significant</th>
<th>Right anwers rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5.4 min</td>
<td>1.7 min</td>
<td>yes</td>
<td>89% 89% 78%</td>
</tr>
<tr>
<td>B</td>
<td>8.1 min</td>
<td>4.0 min</td>
<td></td>
<td>89% 61% 22%</td>
</tr>
</tbody>
</table>

The general comments we got were that the multi-run plot and the coordinated views were considered very useful. Also, the subjects felt that a longer training session would have helped them to further increase their performance with our tool.

Based on the results and the participants’ feedback, we can conclude that our approach and its visual representation are user-friendly and can be successfully applied for a multi-run data analysis by users without any experience and independent of their level of education and background.
6.10 Discussion

6.10.1 Quasi-Monte Carlo Sampling

The first question we would like to address is about the accuracy of our approach when estimating isocontour distances based on the qMC approach. It is well known that the error of an approximation by a qMC method is $O \left( \frac{(\log N)^m}{N} \right)$, where $m$ is the dimensionality of the space and $N$ is the number of random points. However, we do not want to evaluate the error of the area integration, but the accuracy of our distance computation. In other words, we want to estimate how many random points are needed to successfully apply our approach. Hence, we want to analyze the convergence of the distance matrix calculation depending on the number of random points. Using the presented data sets, we computed distance matrices of size $256 \times 256$ for increasing number of random samples. Starting with 256 random numbers, we iteratively increased the number by another 256 random points. After each iteration, we compute the differences ($l^2$ distance) between the current distance matrix and the distance matrix obtained by the previous iteration step. These differences are shown in the plot of convergence in Figure 6.19. We can easily see that we could have also used less than the chosen 32,768 random points.

![Figure 6.19: Convergence plot showing the differences between distance matrices with increasing number of random points. The plot has been generated by constantly increasing the number of random points by 256 and computing the differences between the current and the preceding iteration.](image)
for the climate simulation data set. For example, the 8,196 random points we used for the astrophysical simulation would also have sufficed. Of course, the convergence plot is data dependent, but we made similar observations for all applications we presented.

There are a few factors that affect the choice of the number of random points. When a distance matrix is computed, the amount of memory required to store the information conveyed by the points is $M_{\text{req}} = N \times P$ Bytes, where $N$ is number of compared isocontours and $P$ is number of random points. As $M_{\text{req}}$ grows linearly with increasing $P$, $M_{\text{req}}$ fits the system’s memory even for hundreds of thousands random points. On the other hand, high values of $P$ mean a lot of interpolations, which could be relatively cheap, e.g., when using bi- or trilinear interpolation, or more expensive, e.g., when applying SPH interpolation. Also, when comparing simple isosurfaces representing stars less points are needed than when comparing complicated sea ice isocontours. Therefore, the choice of the number of random points depends on the hardware constraints, on the interpolation computation time, and on the shape of the data features. In conclusion, the user has to choose a proper number of random points being guided by the approximation error of the qMC approach (cf. [89]) and the listed factors. An empirical approach is to perform a scalability test as the one shown in Figure 6.19.

6.10.2 Temporal Subsampling

Another factor which affects the accuracy of the approach is the number of considered time steps. In our examples, we did not make use of all time steps (except for the synthetic dataset). The main issue here is that when using a GPU with 2 GB of memory for the calculations, it is only possible to project a distance matrix which consists of up to approximately $10,000^2$ values (as some memory is reserved by the operating system and other running applications). Thus, we downsampled in the temporal dimension by choosing the number of time steps taking into account the number of considered runs and isovalues. Still, the number of time steps was enough for a fair reflection of the simulated processes. It is commonly possible to downsample in the temporal dimension due to temporal coherence, but the level of downsampling is, of course, based on the underlying simulation set-up. One could also do all computations without using the GPU, which would then allow us to use all provided time steps, but computation times
would be high. For our applications, downsampling in temporal dimension was possible without significantly affecting the results. Technically, there are no restrictions for a distance matrix size.

6.10.3 Dimensionality

The next question we would like to address is the dimensionality of the data. When projecting high-dimensional data to a 2D or 3D visual space, points that were far from each other in the high-dimensional space may happen to be close to each other in the projected space. The intrinsic dimensionality of a dataset can be estimated by evaluating the eigenvalues. We determined the 16 largest eigenvalues for the three considered datasets (synthetic data, gridded climate data, and astrophysical SPH data). The resulting plot is shown in Figure 6.20. It can be observed that the first three eigenvalues are significantly higher than all other eigenvalues. For the climate and synthetic data, even the first two eigenvalues are significantly higher than all others. Hence, we can conclude that for all three datasets using 3D visual spaces allows us to capture the main characteristics of the datasets. For the astrophysical simulations, the third dimension was necessary to successfully capture the characteristics of the polylines (cf. Figures 6.10 (a) and 6.11).
However, we have also observed that for the synthetic dataset it made sense to use the third dimension (cf. Figure 6.6). For the climate simulation the 3D multi-run plots did not exhibit much more information.

### 6.10.4 Investigating Projection Dimensions

Analysis of the projection dimensions is very important, because despite of the low magnitudes of corresponding eigenvectors, it is still possible that such dimensions reflect influence of some data features. Thus we investigated this in more detail. Indeed, it is possible, that some dimensions of the MDS projections represent data features, which dominate the projection output. Hence, less prominent patterns may not be clearly visible in the projected space. Since those less prominent patterns can be important, we performed a more detailed analysis than we had provided. More precisely, we propose to extract more than dominant eigenvectors (i.e., not only the eigenvectors with respect to the 2 or 3 largest eigenvalues, but those up to the 16 largest eigenvalues) and build a similarity plot matrix (in analogy to the scatterplot matrices) of the corresponding 2D projections. The similarity plot matrix for the SPH data set is shown in Figure 6.21. Before, we pointed out that for the SPH simulation data the rotational pattern is dominant. One could invest efforts to exclude the rotational pattern from the data in a pre-processing step. Instead, however, we can also observe from the similarity plot matrix that some dimensions (1, 6, and 11) represent features without rotation. In particular, the projection for dimensions (1, 6), see Figure 6.12 (a), is qualitatively equal to the projection we get when excluding rotation in a pre-processing step, see Figure 6.12 (b). Note, that other dimensions also exhibit clear structural patterns, which means that they also reflect some data features. However, these features are not necessarily different, because they are multi-dimensional and we navigate over their projections. Basing on the obtained results we can conclude that our approach is capable to capture main data features in the extracted dimensions, which can be examined visually using our plots.
Figure 6.21: Scatterplot matrix of MDS projections for the SPH example (45 astrophysical simulations) with 120 different combinations of 16 first biggest eigenvectors. The views corresponding to pairs (1;2), (2;3), (1;6) are presented in the paper.
6.10. DISCUSSION

Figure 6.22: An isosurface similarity map for a randomly selected time frame of the climate simulation data (a) (sea ice thickness), and three MDS projections (b-d) for the selected 1-year simulation for the corresponding isovalues (1.25, 1.5, 1.75). The isovalues have been selected from the solid dark square (red frame). Easy to see, that the differences between the projections are minor.

6.10.5 Robustness of Isosurface Similarity Plots

Sensitivity of the choice of an isovalue for a given data set is reflected numerically and visually in the isosurface similarity maps. Since our similarity measure is based on the volume enclosed by the isosurface (i.e., by all isosurface components), our measure is not sensitive to topological changes of the isosurface. We also investigated the robustness of the resulting plots. Changing the isovalue slightly, i.e., within the range of a block in the isosurface similarity map, leads to a qualitatively similar plot, Figure 6.22. Of course, the plots are only unique up to rotation and mirroring.

6.10.6 Computation Times

To estimate the performance of the proposed approach, we measured the times needed for the distance matrix (DM) and projection (MDS) calculations. For the numerical tests a desktop PC and a laptop were used. The PC was equipped with an Intel Xeon E5 processor, an NVIDIA GeForce GTX 680 GPU, and 8 GB RAM. The laptop had an Intel i7-3630QM processor, an NVIDIA GeFoce GTX 660M, and also 8 GB RAM. The computation times and corresponding parameters for the discussed examples are shown in Table 6.2. The data loading process became one of crucial points for the
DM computations. Despite of its powerful GPU, the PC has a slower CPU and lower system memory capabilities. Therefore, its performance is weaker than the laptop when data loading time exceeds calculation time (see the first and the third row of the table). However, building the distance matrix is performed on the GPU, which reduces the performance of the laptop (see the fourth and the fifth row). The reported projection computation times operated on the GPU are as expected. When the pre-computation is complete, the visual analysis steps can be performed interactively.

Table 6.2: Computation times for pre-processing steps of distance matrix and projection calculation for multi-run data sets of synthetic data, astrophysical SPH data, and gridded climate data.

<table>
<thead>
<tr>
<th>Data</th>
<th>Original resolution</th>
<th>DT resolution</th>
<th>Random points</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>64x64x64</td>
<td>32,768</td>
<td>qMC (GPU)</td>
</tr>
<tr>
<td>Figure 3</td>
<td>64x64x64</td>
<td>64x64x64</td>
<td>32,768</td>
<td>0.41 s</td>
</tr>
<tr>
<td>Figure 4</td>
<td>128x256x256</td>
<td>32x64x64</td>
<td>32,768</td>
<td>0.48 s</td>
</tr>
<tr>
<td>Figure 5</td>
<td>39,718 particles</td>
<td></td>
<td>32,768</td>
<td>0.64 s</td>
</tr>
</tbody>
</table>

6.10.7 Sensitivity of the Approach

The convergence rate of a Monte Carlo approach is well documented in literature on numerical methods. Nevertheless, we have had executed experiments on the convergence for the given application of computing the Jaccard distance, see Figure 6.19. The experiments show that one retrieves already very good results with relatively few samples, see the discussion in the paper.

The downsampling in the temporal dimension depends on the temporal resolution of the simulation. Commonly, the simulation output does not even store all time frames. Still, we have seen that we can reduce the temporal resolution even further to reduce the computational costs while retrieving qualitatively the same results. Ideally, one would run the analysis on the highest temporal resolution, but for efficiency one may choose to downsample. The decision about how much temporal downsampling is applied (if at all), is left to the user. Given these considerations, we nevertheless performed further experiments to create benchmarking results, see Figure 6.23 and Figure 6.24.
6.10. DISCUSSION

Figure 6.23: Temporal downsampling tests on the same data as in Figure 6.22, sea ice thickness = 1.0. All the tests were performed using 32,768 random points. Each test was done with a different frame step (Step:1 means every available simulation frame was used, Step:2 every second, etc.). Times spent for a corresponding distance matrix calculation were measured. We can observe, that due to temporal downsampling only some fine details of the curve disappear, while the main structure of the curve stays the same. Thus, there is significant benefit in terms of calculation time, without losing the important details.

Figure 6.24: Random points number tests on the same data as in Figure 6.23. All the tests were performed using Step:4. Each test was done with a different number of random points. Times spent for a corresponding distance matrix calculation were measured. We can observe that due to decreasing number of random points, the quality of the projection suffers (the climate simulations have a large number of fine details), the correct structure can be still recognized though (except of (d)). There is no significant benefit in terms of calculation time.

Concerning the scalar value comparison to determine interior of isosurfaces, the error of the approximation depends on the interpolation approach. For example, for the SPH simulations we use the same interpolation kernel that is used during the simulation. Hence, our inside-outside test is consistent with the simulation. For the gridded
climate simulations, we use bilinear interpolation to reconstruct the field. This is a standard interpolation method for the visualization of gridded data and widely accepted in the community. When explicitly extracting an isocontour, one would need to apply the same interpolation methods. Our method is even more accurate than, for example, the Marching Squares approach, where the isocontour within each grid cell is approximated by a straight line, while our approach actually uses the implicit isocontour representation with respect to bilinear interpolation. In general, our method is consistent with isosurface extraction methods. There is no additional approximation error introduced.

6.10.8 Visual Complexity

We acknowledge that the visual complexity of the plots is something that needed further discussion. The complexity is impacted by the number of selected isovalues ($N_{iso}$), the number of given simulation runs ($N_{runs}$), and the number of time steps for each simulation run ($N_{steps}$). Our visual encoding rendered ($N_{iso} \times N_{runs}$) polylines with ($N_{steps}$) vertices. Hence, the visual complexity is determined by those three parameters, which we would like to comment on.

*Number of selected isovalues*

This part is related to the selection of appropriate isovalues. We also discuss this issue in this regard:

a) In most application domains, one can assume that the domain scientists know very well the range of the simulated function values and know what would be an isovalue of interest that should be picked. For example, climate scientists investigate sea ice distribution with a certain value of ice concentration. Hence, selecting a single isovalue is often enough ($N_{iso} = 1$) and this is done interactively using the domain knowledge of the user.

b) In case a suitable isovalue is not known, there exist methods to analyze scalar fields for finding interesting isovalues. A widely accepted approach in the visualization community is the usage of the isosurface similarity map as proposed by Bruckner and Möller [9]. We have followed this idea and showed that isosurface similarity maps
can also be generated using our isosurface similarity measure, which has a number of advantages, see Chapter 3. Using this visual representation of isosurface similarity, a suitable isovalue can be selected for further investigation.

**c)** Even without the usage of isosurface similarity maps, one can determine possibly interesting isovalues using directly our proposed methods. To do so, one would sample the range of the scalar field (e.g., equidistantly) and render the polylines for a subset of selected simulations. This has been documented in the Sections 6.4 and 6.5. It can be directly seen from the provided visualization, which isosurfaces change significantly in the ensemble (and are potentially interesting) and which are not varying much. One would then select an isovalue with significantly changing isosurfaces.

**d)** Obviously, the solutions in b) and c) can be combined such that the isosurface similarity map provides candidates (instead of sampling the range equidistantly), which are then visualized using the proposed similarity plot to select one isovalue from the candidates.

**Number of selected time steps**

In simulations, the provided time intervals are typically small leading to many time steps. A downsampling in the temporal dimension is feasible to reduce computational costs and memory requirements. With the temporal downsampling rates reported before, the plot itself would not change much qualitatively. On the other hand, the visual complexity is not reduced and the polylines may exhibit some artificially introduced sharp turns. To improve the visual quality especially in the temporally downsampled plot, we replace the polyline with a smooth curve rendering. The smooth curve is generated using cubic spline interpolation. The results are visually appealing and the removal of artifacts even reduces the visual complexity, as similar curves stay closer together (no corner cutting).

**Number of simulation runs**

The objective of the paper is, of course, the visualization of the entire multi-run data set. Hence, all runs shall be considered. However, there are several considerations to this:
6. MULTI-RUN SIMILARITY PLOTS

a) Since similar runs lead to similar curves in the similarity plot, the curves clutter. However, this is exactly what one wants to see. It is of interest to observe this group of similar curves that come together, where all runs have similar time steps, and deviate for other time steps. To document the visible patterns, we consider matrices of 2D similarity plots for the various data sets for a pairwise combination of the most relevant principle directions. When investigating these plots, one can observe that there are obvious patterns in the data set. Moreover, outliers can then be easily spotted.

b) Another possible way could be aggregation of curves to reduce the clutter. But a bundling strategy could introduce artifacts. For example, if the curves are parallel to each other with equal distances between them, one can conclude that there is a smooth transition over the runs. A bundling would destroy this insight and leading to the illusion that there are groups of similar runs represented by the bundles. Hence, bundling can lead to misinterpretations and we believe that bundling is not the right strategy here. Another way is to aggregate the rendering. We followed this way by producing similarity plots, where density is mapped to an RGBA color using a transfer function, see Figure 6.9 (b). We showed that such a visual representation can help to see subpatterns or outliers, which may have been occluded by main patterns otherwise. Hence, it provides a good overview. However, it is not so suitable for drilling down for the analysis of individual runs or for a comparative analysis of a few selected runs. We added this density-encoding representation as an option to our system and allow for interactively switching it on and off.

c) Our approach is a combination of visualizations and appropriate interaction tools. Hence, showing all runs in one plot is only the starting point for further analysis, following the well-known Shneiderman mantra overview first, zoom and filter, then details-on-demand. The first plot of all runs gives an overview. We provide methods for selection and filtering to focus on those runs that shall be further investigated. Then, only those will be shown (possibly using different rendering styles in terms of line thickness and different encodings in terms color mapping) and we can use different interaction methods to dig deeper in the analysis using re-configurations of the plot and coordinated views. Of course, we can also restrict the analysis interactively to a certain time interval, which is another zooming mechanism.
6.11 Conclusion

We presented a novel isocontour-based approach to visually analyze multi-run spatio-temporal simulation data independently on their structure and spatial dimensionality. The approach is sensitive to shapes of isocontours which serve as a descriptor of the scalar fields. Therefore, it is possible to effectively capture all information about the simulations instead of just using means or other statistical data. We introduced a distance function for comparing isocontours, which allows us to apply projection methods to generate a distance (or similarity) plot. We have analyzed and documented the benefits of our proposed isocontour distance computation. Using MDS projection, we have shown that our approach can be successfully applied for the visual representation of ensembles, for grouping simulations by behavior pattern, for outlier detection, and for the analysis of the influence of initial parameters to the evolution of the modeled system. In combination with a linked view to physical space visualizations, our approach supports fast extraction of information about a state of the simulated system for any time step. We have performed tests to show that the proposed approach is applicable equally well to simulations from different fields of science involving different types of data. We also conducted a user study to show the benefit of using our visual analysis tool. A parallel implementation allows us to run our methods on standard PCs and laptops.
Chapter 7

MultiVisA tool and application scenarios

This chapter was prepared to be submitted as a full paper:


Physical simulations aim at modeling and computing spatiotemporal phenomena. As the simulations depend on initial conditions and/or parameter settings whose impact is to be investigated, a larger number of simulation runs is commonly executed. Analyzing all facets of such multi-run multi-field spatio-temporal simulation data poses a challenge for visualization. It requires the design of different visual encodings that aggregate information in multiple ways and at multiple abstraction levels. MultiVisA is a tool for the interactive visual analysis of multi-run data from physical simulations based on a number of aggregated plots and coordinated interactions. A histogram-based plot allows for the investigation of the distribution of function values within all simulation runs. A density-based time-series plot allows for the detection of patterns and outliers within the ensemble of multiple runs for single and multiple fields. A similarity-based plot allows for the comparison of multiple or individual runs and their behavior over time. Coordinated views allow for linking the plots to spatial visualizations in physical space. We apply MultiVisA to physical simulations from the field of climate research
and astrophysics. We document the analysis process, demonstrate its effectiveness, and provide evaluations involving domain experts.

### 7.1 Introduction

Simulations of time-varying phenomena over a 2D or 3D spatial domain are widely used in the field of physics (among others) to test the respective mathematical or computational models. The simulations typically depend on a number of parameter settings or initial conditions. Since one of the research tasks is to understand how the input settings influence the simulation result, the simulations are run multiple times with varying settings. Thus, researchers gather multi-run spatio-temporal data with many runs and many time steps, where each time step of each run represents planar or volumetric data fields. The analysis of such a data set raises the challenges of efficiently handling the large amount of data and effectively comparing the outcome of multiple simulation runs. Since it is not feasible to analyze all time steps of all runs individually, one needs to aggregate information about the entire ensemble of simulation runs.

Currently, in research communities dealing with simulation ensembles, there is the lack of a unified approach for processing, navigation, feature detection, and comparative analysis of entire ensembles. It is common practice that researchers develop their own ad-hoc solutions to their analysis tasks by developing scripts that stitch together existing tools for solving subproblems. Visualization methods are typically only used for the rendering of phenomena in physical space, i.e., at the very end of the analysis process. In this paper, we present MultiVisA, an approach to the interactive visual analysis of multi-run spatio-temporal physical simulations that supports a top-down analysis process of entire ensembles.

MultiVisA is based on three types of aggregated plots linked with physical space visualizations and a portfolio of interaction mechanisms. The plots intuitively provide comprehensive information of the simulation ensemble at different aggregation levels. The field distribution histogram aggregates field value occurrences over all time steps and all runs. This first overview allows the user to identify the relevant data range for further analysis. The function plots aggregated over all runs support multiple analysis
steps related to time series. First, they allow for the detection of relevant time steps and the synchronization of features in multiple runs. This feature detection and selection step restricts the subsequent analysis steps to the relevant time intervals, which often reduces the amount of data to be analyzed tremendously. Second, the function plots intuitively depict behavioral patterns over time. The governing patterns and outliers within the ensemble or within individual runs can be detected. And, multiple coordinated function plots allow for an intuitive comparative analysis of multiple fields. Finally, the function plots exhibit the range of activity, which allows the user to identify representative isovalues for further analysis. This further analysis is supported by the multi-run plot, which is a similarity plot based on isocontour similarity of different time steps of different runs. Hence, it allows for a comprehensive understanding of the entire ensemble of simulation runs by depicting each of them as a polyline, where divergence or convergence of the polylines indicate how much simulations differ over time. Our plots are incorporated in one interactive analysis tool using coordinated views, which includes brushing and linking to physical space renderings.

The visual encodings and interaction mechanisms provided by MultiVisA are described in Section 7.2, while Section 7.3 is dedicated to documenting how MultiVisA is applied to a top-down analysis of physical simulations. We chose two application scenarios of quite different data characteristics. The first application provides multiple runs of astrophysical smoothed particle hydrodynamics (SPH) simulations over 3D point-based spatial domains, where the runs differ by setting different values of the input parameters. The second application provides ensembles of climate simulations over 2D gridded spatial domains with a set of different initial conditions. We show the effectiveness of our analysis tool by documenting the processing pipeline of our approach, discussing the findings that can be obtained at the various analysis stages, and reporting the feedback from domain scientists.

For the related work about multi-run multi-field spatio-temporal simulation data analysis we refer to Chapter 2.

Despite the availability of existing techniques, most researchers who are trying to analyze their ensemble simulations spend days or weeks to prepare and analyze simulated data for further analysis. Usually they implement their own scripts (customized to
7.2 VISUAL ANALYSIS OF MULTI-RUN SIMULATION DATA

Since the main purpose of executing multi-run simulations is to capture the variety in the model with respect to different initial settings or parameter selections, an ensemble can consist of tens or even hundreds of simulations. Despite the same nature of all runs within an ensemble, their outcome may have high dispersion that needs to be investigated. Independent of the simulation method (Eulerian or Lagrangian), the spatial data structure (gridded or point-based), and the purpose (impact of simulation parameters or model evaluation), the visualization tasks can be identified as (1) defining visual encodings in the form of plots that exhibit the proper level of aggregation and (2) defining interaction methods for operating on differently aggregated plots and physical space renderings using coordinated views.

For the development of a successful analysis tool, several characteristics of multi-run spatio-temporal simulation have to be considered. First, the data size frequently exceeds hundreds of Gigabytes, i.e., the data set does not fit into the main memory of a system. Thus, every access to the entire data is extremely time consuming. Even simple computations such as computing the mean can take up to hours. Hence, aggregated information plays an important role and being able to concentrate on a region of interest (part of the data) can substantially reduce the computational load. Second, due to the multiple facets of multi-run data [5], different representations are required to shed light on different aspects. Finally, it is of interests to compare the simulations’ behavior and evolution over time, which is complex task due to the large number of simulation runs. Computing means is often not sufficient, as behaviors of individual runs may not be
reflected anymore.

Having pointed out the challenges we are facing, the analysis of multi-run spatio-temporal data can be executed according to the following workflow:

1. **Overview analysis of field range distribution.** In a first stage, one is interested in getting an overview of the ensemble, which can be achieved by investigating the range of the considered data field and the distribution of field values within the simulation runs. Respective histograms allow for first conclusions and to narrow down the field range for subsequent analysis stages (see Section 7.2.1).

2. **Analysis of field distribution over time.** In this stage, one would like to investigate the change within the simulation runs over time, which supports multiple important tasks. First, one can detect features and the time intervals they occur, which narrows down the time interval for further analysis steps. Second, one can identify individual field values of interest, which can be further examined, e.g., by choosing them as isovalues. Third, one can detect overall patterns in the ensemble as well as outliers. A run identified to be of interest can also be observed individually as well as in further analyses with physical domain visualizations. Finally, one can also compare and correlate different fields of a multi-field data set at this level (see Section 7.2.2).

3. **Comparative analysis of individual runs.** While the second stage was operating on an aggregation over multiple runs, this stage shall allow for a detailed understanding of the behavior of individual runs in a comparative view. Making appropriate selections in the preceding stage (i.e., identifying time interval and field value of interest) allows for an accurate and efficient analysis approach (see Section 7.2.3).
7.2 VISUAL ANALYSIS OF MULTI-RUN SIMULATION DATA

7.2.1 Field Distribution Histogram

Assuming that the data to be analyzed have not been studied yet and the simulation results are still unknown, we propose to start with a simple overview plot based on the estimation of the range of the investigated data field and the analysis of the probability distribution of the occurrence of the field values. This step allows us to detect simulations with outstanding field values and to define the main global data features such as global field range, shared field range (i.e., the intersection of the ranges of all simulation runs), values with high and low frequencies of occurrence.

The visual encoding is implemented by building a histogram with field values on the horizontal axis and normalized frequencies of occurrence on the vertical axis. The histogram aggregates information from all points in space and time for all simulation runs. The field values from the intersection of the ranges of all time steps of all runs are colored in green, the values from the intersection of ranges of all runs (but not from all time steps of the runs) are colored in blue, and values that do not occur in all runs are colored in red, see Figure 7.1.

The interaction mechanisms that support the analysis allows for the selection of individual field values (using a vertical line), which reports back all simulation runs where this field value occurs, which is particularly useful for investigating outliers. Moreover, the user can select a field range for further investigations by cutting intervals to be neglected, which narrows down the analysis to a region of interest.

Figure 7.1: Field distribution histogram (for astrophysical simulation). Field values from the intersection of all time steps’ ranges of all runs are colored in green, from the intersection of all simulations’ ranges (but not for all time steps) in blue, otherwise red.
7.2.2 Function Plot

At the next analysis stage, we aim at investigating change over time. We propose to use function plots that record how the field values at the spatial data samples vary over time for the simulation runs. The plot represents the function values of each spatial data sample of each simulation run as a piece-wise linear graph of a time series. For the visual encoding, we aggregate the time series lines over a 2D grid leading to a 2D density histogram (effectively aggregating over spatial positions and simulation runs). Then, we can apply a transfer function to map the accumulated density values to color. An example of a function plot is shown in Figure 7.2 (top right) when applying the transfer function shown in Figure 7.2 (top middle). We use this transfer function throughout the paper. Note that the transfer function is applied to the range of interest that was selected using the field distribution histogram, i.e., the selection in the field distribution histogram makes the visual representation of the function plot more effective.

Function plots (or time histograms) have been used before for time-varying scalar fields [22, 90–92]. We extend their application to visualize ensembles of simulations. Moreover, we want to point out that the underlying data structure is that of piecewise linear curves that represent time series. Consequently, we do not generate static his-
tograms, but can perform interactions on our plot. More precisely, we can brush on the function plots to interactively select all curves that traverse a selected region of interest and interactively update the plot to only render the aggregated selected curves (see Figure 7.12). Furthermore, we can interactively switch between aggregating over all runs, a selected subset of runs, or individual runs. When rendering function plots of individual runs, brushing on the plot (see Figure 7.4) triggers linked physical space visualizations of the selection (see Figure 7.5). Also, when observing multi-fields, we can produce one function plot per fields, compare and correlate them with each other, and have coordinated brushing and linking between the multiple function plots (see Figure 7.12). Finally, we can also select a specific region of interest for further analysis. In particular, when detecting a feature, one can cut the time axis to a time interval that contains the feature, which makes the subsequent analysis steps more efficient and effective. Also, we can select a field value for further analysis purposes (based on similarity plots, see below) using a horizontal line (see Figure 7.8).

Since we are typically dealing with a large amount of runs with a high spatio-temporal resolution, we have to accumulate many curves with many time steps for the generation of a plot. To allow for their generation at interactive frame rates, we use a level-of-detail representation of the curves coupled with progressive rendering. The level-of-detail approach uses a hierarchical representation based on 1D Haar wavelets. The progressive rendering approach accumulates all curves first at their coarsest resolution and refines the representation iteratively until the finest resolution is reached.

### 7.2.3 Similarity Plot

In our third stage, we want to generate a visual encoding that allows us to perform a comparative analysis between the runs of an ensemble. Hence, we should not anymore aggregate over the runs. The idea of the proposed approach is to use time lines in a similarity plot (or multi-run plot), where the similarity is measured by looking at (2D or 3D) isocontours of individual time steps. Isocontours are known to be effective field descriptors and can capture the simulation states within the physical domain for the runs at each point in time. For the details and discussion of the concept of multi-run similarity plot we refer to Chapter 6.
We also support a number of interaction mechanisms on the similarity plot. First, instead of showing all runs, we can show a subset or even individual ones (see Figure 7.10 (c)). Also, parts of the plot can be selected and a new projection of the selected part can be generated. Since the precomputed similarity matrix can be re-used, this remains interactive. In particular, we can select one time step such that the time lines reduce to points (see Figure 7.10 (b)). We can also select individual points on the time lines to trigger a physical-space visualization either in a coordinated view (see Figure 7.9) or in an embedded view (see Figure 7.10 (b)). Furthermore, we allow for switching between projections to 2D and 3D visual space using two or three principal components. Alternatively, one can only use one principal component as a vertical axis in a 2D plot, where the horizontal axis represents time (see Figure 7.10 (a)).

7.3 Case Studies

7.3.1 Astrophysical Simulations

To test the effectiveness of MultiVisA for the analysis of multi-run physical simulations, we executed two case studies, where we apply the methods and workflow as described above. The first case study is concerned with an astrophysical two stars system of White Dwarfs. The ensemble consists of 45 simulations with two main parameters representing the masses of the two stars. Each simulation run consists from 400 to 1,300 time steps. Overall this data set contains about 36,000 time steps, which sums to approximately 170 GB of data.

Stage 1 - Field Distribution Histogram. We start our analysis by computing the field distribution histogram for the scalar field of Internal Energy as shown in Figure 7.1. It is a simple plot, but nevertheless allows for some first interesting observations: (1) The distribution is skewed towards the lower values. In fact, only very few values are populating the upper half of the histogram. The respective simulation runs can immediately be identified as outliers by selecting the respective regions in the histogram. (2) After having identified the outliers, further analysis steps shall be applied to a narrowed (more saturated) field interval that excludes the outliers. This will make the automatic appli-
Figure 7.3: Function plot of astrophysical simulation of two stars both with masses equal to 1.05 of the solar mass. Time steps around 300 contain outliers in field values and exhibit a significant change in the simulation structure, while before and after this change almost steady patterns can be observed.

Stage 2 - Function Plot. In the second stage, we operate on the function plots. Figure 7.3 shows the function plot for a single simulation run that was identified as an outlier in Stage 1. In this simulation run, both stars have the same mass. To observe the outlier values, we did not apply the narrowing of the field range from Stage 1. We can observe that there are very few field values with an internal energy greater than 3.0 and that they occur around time step 300. Selecting those outliers and investigating them in a coordinated physical space visualization, one can observe that they belong to particles that transition from one star to the other. When hitting the other star the internal energy of these particles suddenly rises to high values, but also very quickly drops down again.

Apart from the investigation of the outliers, Figure 7.3 clearly indicates that we can distinguish three phases during the simulation. First, there is relatively steady state up to a short period, where things are changing (around time step 300), which is followed by another relatively steady state. Looking at other simulation runs from the same ensemble, we can observe a similar behavior pattern consisting of three phases, but the distributions of field values during the simulations are different. Figure 7.4 shows the function plot for a simulation run where one of the stars is much larger than the
7. MULTIVISA TOOL AND APPLICATION SCENARIOS

Figure 7.4: Function plot of the astrophysical simulation of two stars with different masses equal to 0.65 and 1.05 of the solar mass. Also the same three phases can be distinguished as in Figure 7.3, but additional feature can be observed for a hot matter. Interactive selection (shown in green) for coordinated view to the linked physical domain visualization.

other (now the field range is cropped according to the observations in Stage 1). When comparing Figures 7.3 and 7.4, we can observe an additional feature. To further analyze this, we again brush on the plot and investigate what corresponds to those features in a coordinated physical space, see Figure 7.5. We observe that initially we have two stars (a) in the first phase. In the second phase, for the heavier one we have a slightly increasing internal energy (b), while the lighter star loses its mass and internal energy (c). At these time steps when the stars are merging, the function plot allows us to easily and clearly separate the matter of the two stars according to their field values. Finally, the lighter star is completed absorbed by the heavier star in the third phase (d).

As mentioned above, the three phases occur in every simulation run. Moreover, the initial and the final phase are pretty static. Not much is happening there. Indeed, from an astrophysical point of view, the transitions between the phases is of interest. Hence, one can crop most of the initial phase and the final phase without losing valuable information. Using our function plots, we can interactively cut the simulation runs to a small time interval that fully captures the merging phase and only the end of the initial phase and the beginning of the final phase. Hence, it still includes all transitions. Identification of such time intervals for multi-run simulations is crucial and usually takes a significant amount of time. With our tool, it is possible to visually identify the time intervals and manually crop to the desired time interval.
Figure 7.5: Linked views of selections in the Figure 7.4 in physical domain. Selection (a) represents two separated stars. Selection (b) shows that the shell of the core of one star is in the same condition, while selection (c) shows that the matter of the other star is absorbed by the first one. Selection (d) shows the merged structure. Note that the representation of the heavier star seems smaller, as it represents data points with higher internal energy and therefore is only a core.

Using side-by-side comparisons, we can intuitively compare the function plots of two simulation runs. When trying to get an overview of the entire ensemble, we proposed to aggregate the information of all runs in one plot. When comparing the two plots in Figures 7.3 and 7.4 we observe that the merging phases occur at different time steps during the simulation. This lies in the nature of the simulation, as the runs are not synchronized and different runs even have different amount of time steps. Thus, when aggregating all 45 simulation runs without synchronization, we cannot observe any general pattern, see Figure 7.6. Using the manual cutting of our function plots as described above, a manual synchronization of all runs is intuitively possible. Although this just requires a single selection for each run, we still need to go through all plots once. Hence, we considered that it may be useful to automate this step. The idea is to synchronize the plots by the merging phase and to apply a respective shifting. The advantage of our method is that such features can be estimated using image-based processing. We simply identify the highest gradient of the plot density, as it is reached in the merging phase. Since the duration of this phase is different for the runs, we take the center of the time interval of 20 time steps with the highest sum of gradients as our synchronization point. Figure 7.7 shows the function plot aggregated over all simulations runs after automatic synchronization. Now, we can also clearly observe the three phases in this function plot.
7. MULTIVISA TOOL AND APPLICATION SCENARIOS

Figure 7.6: Function plot aggregated from all 45 astrophysical multi-run simulations without synchronization. General structure cannot be recognized. Vertical discontinuities indicate ends of simulations.

Figure 7.7: Result of automatic function plot synchronization for the astrophysical multi-run simulation data. As opposed to the unsynchronized representation in Figure 7.6, details of the general structure (three phases) can be observed.

Another feature of our function plot approach is that it is not restricted to create this one representation of the data field, but we can also derive further field. For example, when taking the function plot aggregated over all synchronized runs as the average mean, we can compute a plot representing the standard deviation. Figure 7.8 shows the result of such a computation. We can see the benefit of such a standard deviation plot. While the plot in Figure 7.7 did not exhibit strong differences, the standard deviation plot exhibits more clearly visible structures. Despite the similar structure of the field distribution in all the simulation runs over the simulation time, we can observe that the highest deviation is present in the lowest field values. The reason is that in every simulation run there is the same total number of data points, but describing different simulation states the proportion of data points representing the considered field range is
different. It means that a more detailed comparative analysis is required to investigate, how the simulation states differ in terms of physical structure. To do so, we proceed to Stage 3 by choosing a representative field value, i.e., a field value which describes best the important data features in each individual simulation run. Selecting the field value around 0.3, as shown by the green horizontal line in Figure 7.8, we cover the main part of the structure with the highest deviation for all three simulation phases.

**Stage 3 - Similarity Plot.** Having cropped the time intervals in Stage 2, which significantly reduces the amount of data to be handled, and having identified a representative field value, we can make use of that field value as the selected isovalue for the isosurface similarity computation. Having computed the isosurface similarity matrix, we generate the similarity plot. For the given application, we decided to generate a 3D similarity plot, which can be visually inspected using rotation and zooming. Figures 7.9 (a) and (b) show two orthogonal views on the 3D plot. It shows all 45 astrophysical simulations. The polylines are color-coded using a continuous transfer function that maps the simulation parameter of the star’s mass to the brightness of the color. Increasing mass leads to decreasing brightness. We can observe a clear structure in the 3D similarity plot. Figure 7.9 (a) confirms the finding from Stage 2 that we have three simulation phases. For the similarity plots, we can use coordinated views to physical space rendering when displaying all runs, as the information is not aggregated in the plot. Thus, when selecting a point in the plot, the field of the respective time step of the respective run is displayed in
Figure 7.9: This figure is taken from Chapter 6 (Figure 6.11) as a reference to the interactive MDS projection investigation. Two orthogonal views of 3D similarity plot (a, b) and selected keyframes, which are displayed in linked views to physical domain visualization (c-f).

Since we are displaying all simulation runs together, it is obvious that the visual complexity of the plot increases with increasing number and duration of the simulation runs. The similarity plot including all geometry is meant to give an overview and exhibit
7.3. CASE STUDIES

(a) Projection, by time

(b) Projection, one time step

Figure 7.10: (a) Plotting principal component of projection (vertical axis) over time (horizontal axis). (b) Similarity plot for one selected time step with embedded physical space visualizations. (c) Similarity plot of one selected simulation run after cropping time interval. (d) Similarity plot of same selected simulation run for the full time series but skipping every other frame, which leads to down-sampling artifacts.

Another option we provided was to plot the first principal component of our projection against the time dimension. Figure 7.10 (a) shows such a plot. Now, we can easily see that increasing mass leads to a shift of the lines to higher positions in the projection. Hence, there is a straight dependence between initial parameter and a simulation behavior, which is documented by the continuous color transition in this plot.
To investigate this phenomenon, we propose to operate with the projections interactively. We can select certain parts of the similarity plot and recompute the projection only considering the selection, i.e., not selected points do not affect the projection result. Figure 7.10 (b) shows a recomputed projection for a selected single time step from the end of simulation. The darkest points are located in the upper right corner and the brightest in the lower left corner. In between, there is a color transition visible. To correlate that to physical space, we chose the option of embedded views, i.e., the physical space visualizations of selected points are embedded as small icons in the similarity plot.

**Domain Expert Feedback.** We discussed the MultiVisA tools and its components with the domain expert who generated the data set. We asked for advantages and limitations of our approach and to comment on the effectiveness or usefulness of our approach. The main findings were:

- To identify simulation features within a whole ensemble, researchers are using their own scripts and subroutines, as even advanced applications such as SPLASH [93] do not provide enough functionality. With our tool a multi-run simulation analysis becomes easy to visualize and it allows for a faster data investigation.

- The task of time alignment is one of the most time consuming for the researchers. To perform the alignment on the whole ensemble of 250 simulations one needs to spend couple of weeks, while with our tool one can do it by a single click.

- Correct and precise time definition of the analyzed features leads to increased accuracy of the analysis steps. For example, one can significantly increase the quality of the MDS projection when narrowing down the time interval. Figure 7.10 (d) shows the same similarity plot as in (c), where in (c) one could use all time steps within the shorter time interval, while in (d) one could only use every other time step of the full time series.

- The domain expert has been working on this data set for a long time and knows it very well. Using our tools he was able to recognize most of the known data features in one session. Moreover, he even identified some additional features for further investigation.
7.3.2 Climate Simulations

In the second application scenario we investigate an ensemble simulation using a global climate simulation model over one to three years with different initial conditions. The 11 simulation runs have a duration of 1,460 to 4,380 time steps. The simulations start at the same initial time, but are based on different initial conditions. The total data size is 45 GB and includes four different scalar fields: sea surface temperature, sea ice thickness, sea ice concentration, and sea snow thickness.

Computing of the field distribution histogram for sea snow thickness does not allow us to identify any outlier (see Figure 7.11). Hence, we consider the whole field range for further analysis.

As it is of interest to analyze the multi-field aspect, we generate function plots for multiple fields. Since the simulations are synchronized, we can immediately aggregate over all runs. Figure 7.2 shows the function plot for sea ice thickness, Figure 7.12 (1) the respective plot for sea snow thickness. In both plots we can observe the repeating annual pattern, but the overall structure of the plots is different. To investigate the plot in Figure 7.12, we made two selections. We rendered the selected trajectories in the function plot, see Figures 7.12 (2-3) and in a coordinated physical domain visualization, see Figure 7.13 (a, b). We observe that the selections exhibit two annual patterns, which correspond to high snow thickness values for the winter season in the arctic and antarctic region, respectively. Our tool also allows for brushing and linking between multiple function plots. Thus, in Figure 7.12, the selection for snow thickness (second from top)
is transferred to ice thickness (4). We can clearly observe the correlation between the two patterns, yet there are visible differences.

Figure 7.12: (1) Function plot for snow thickness aggregated over all 11 climate simulation runs exhibits annual patterns, which are selected as shown in green. (2-3) Function plots for snow thickness when only showing the trajectories according to selections (a) and (b), respectively. (4) Function plots for ice thickness (as in Figure 7.2) when showing the trajectories according to selection (a).
Figure 7.13: Coordinated views to physical space visualizations of selections in Figure 7.12 exhibit that selection belong to arctic region (a) and antarctic region (b).

Figure 7.14: Plotting principal component of projection (vertical axis) over time (horizontal axis) for all 11 simulations when selecting arctic (a) and antarctic (b) region separately. For isocontour similarity, we considered isovalue 0.25 of sea snow thickness. We observe no activity during summer months in (a), but activity throughout the year in (b). Note that the metric for the distance computation returns absolute values such that both plots are oriented the same way.

We also generate similarity plots for the ensemble, see Figure 7.2 (bottom right). Annual patterns can be observed again, but we also see that the runs differ quite a bit for certain months (to the left), while they are similar for other months (to the right). It is of interest to analyze, where in physical space the differences occur. We use the first principal component of the MDS projection plotted against time and compute the plots considering arctic and antarctic regions separately, see Figure 7.14. One can observe that both regions exhibit a seasonal pattern, but in the arctic (a) there is no activity during the summer season, while in the antarctic (b) there is activity throughout the year.
Moreover, the plot in (b) has higher variance and some outliers, which are candidates for further investigations.

Again, we discussed the application of our tool with the domain expert who generated the data and had the following findings.

- Visualization of the entire ensemble at once allowed him to estimate the diversity of the simulations’ behavior and to identify patterns and outliers.
- A strong advantage is the option to estimate easily activities of subregions. Usually one would need to look at some physical domain visualization for some selected time steps. Our tools leads to increased accuracy of feature detection.
- Estimating the influence of initial conditions to the simulation result is usually performed in sensitivity studies. A large number of statistical descriptors needs to be used. While it is complicated to capture the behavior differences with a single value descriptor, our approach captures them in a multidimensional fashion and allows for interaction and navigation.

7.4 Conclusion

We presented MultiVisA, a visual analysis approach for multi-run spatio-temporal data analysis in the context of physical simulations. We identified the needs of domain scientists to have a visualization tool that supports early steps of the analysis process. MultiVisA uses plots at different aggregation levels to support the analysis workflow in a top-down manner. We applied our tool for case studies in climate research and astrophysics. We were able to perform effective and efficient analyses and got encouraging feedback from the domain scientists saying that MultiVisA can indeed improve their analysis tasks. All the proposed algorithms were efficiently implemented using parallelization on CPU and GPU where applicable, which allowed for a smooth user experience during the interactive sessions using standard PCs or laptops.
Chapter 8

Comparative visualization of multi-run multi-field time-varying spatial data

This chapter was prepared to be submitted as a full paper:

Alexey Fofonov, Lars Linsen. Projected Field Similarity for Comparative Visualization of Multi-run Multi-field Time-varying Spatial Data.

The purpose of multi-run simulations is to capture the variability of the output with respect to different initial settings. Comparative analysis of multi-run spatio-temporal simulation data requires us to investigate the differences in the dynamics of the simulations’ changes over time. To capture the changes and differences, aggregated statistical information may often be insufficient, and it is desirable to capture the local differences between spatial data fields at different times and between different runs. To calculate the pairwise similarity between data fields, we generalize the concept of isosurface similarity from individual surfaces to entire fields and propose efficient computation strategies. The described approach can be applied considering a single scalar field for all simulation runs or can be generalized to a similarity measure capturing all data fields of a multi-field data set simultaneously. Given the field similarity, we use multidimensional scaling approaches to visualize the similarity in 2D or 3D projected views as well as plotting 1D similarity projections over time. Each simulation run is depicted as a polyline within the similarity maps. The overall visual analysis concept can be applied using
our proposed field similarity or any other existing measure for field similarity. We evaluate our measure in comparison to popular existing measures for different configurations and discuss their advantages and limitations. We apply them to generate similarity maps for real-world data sets within the overall concept for comparative visualization of multi-run spatio-temporal data and discuss the results.

8.1 Introduction

The investigation of spatio-temporal volumetric phenomena is ubiquitous in the Sciences and Engineering. Modeling them mathematically allows for computational means to run simulations with varying start configurations. Such multi-run time-varying volume data typically also involves multi-field aspects, i.e., each time step of each simulation run is represented by multiple data fields. Our goal is to analyze the changes over time including different fields in multiple runs. We propose to use the concept of multi-run similarity plots, see Chapter 6, which has been introduced based on the comparison of isocontours. Single isocontours, generally, do not capture the entire volumetric field sufficiently well and finding a number of representative isocontours (if existent at all) is not a trivial task. Hence, we propose to replace isosurface similarity with field similarity in this paper. Moreover, we define a novel field similarity measure and show that it is a generalization of the isosurface similarity measure (Section 8.4.1), where even the efficient computation strategy generalizes. Our framework could, theoretically, be applied with any field similarity measure (Sections 8.4.2 and 8.4.3). We discuss the advantages of our new field similarity measure in comparison to commonly used other measures that can be considered the state of the art (Section 8.5). Finally, we also extend our field similarity measure to a multi-field similarity measure to allow for multi-run multi-field spatio-temporal data analyses. We apply our approach to data from different domains and with different characteristics.

The multi-run similarity plots generate pairwise distances between all time steps of all simulation runs (Section 8.3). Using a multi-dimensional scaling (MDS) approach, the data are projected to a visual space, where simulation runs are represented by polylines connecting the different time points. Depending on the analysis of the eigenvalues
in the MDS approach, we project to a 1D visual space, where values are then plotted over time, a 2D, or a 3D visual space.

8.2 Related Work

Obermaier and Joy [94] recently summarized challenges in simulation ensembles. They point out that the main challenging direction is the visualization and exploration of multidimensional parameter spaces. The raise the question “how highdimensional data visualization techniques can help connect ensemble and parameter-space analysis”.

Many ensemble visualization approaches have recently been introduced. The techniques presented by Phadke et al. [29] are limited to comparing only a small number of ensemble members at any given time. A novel technique for the interactive visual exploration of large 3D scalar ensembles was introduced by Demir et al. [95]. A static ensemble visualization system that automatically helps users locate interesting subsets of members to visualize was proposed by Hao et al. [96]. Both approaches currently have no extension to ensembles which vary over time. Estimation of the uncertainty represented by the simulations within an ensemble was investigated by Pöthkow et al. [30]. An approach for uncertainty-aware multidimensional ensemble data visualization and exploration was recently presented by Chen et al. [97], but it is also does not allow to compare behavior patterns of individual simulations over time. An approach for the analysis of ensemble data set using statistical descriptors was developed by Potter et al. [26]. The feature based ensemble visualizations were investigated by Sanyal et al. [28] and Alabi et al. [98]. To compute statistical properties over the simulation domain location based visualization are used, such as visualizations proposed by Potter et al. [27] or by Hummel et al. [99]. These approaches do not provide exploration methods for the input parameters, i.e., they do not allow the users to compare and group input datasets to pick necessary attributes for further analysis. Also, most of the approaches are based on displaying statistical information, which often does not allow for a comprehensive analysis of the simulation features in a set of multiple runs.
The multi-run similarity plot approach presented in Chapter 6 tackles the aspect of multi-run spatio-temporal data visualization and analysis, which allows for an exploration of both the parameter space and the physical space. Its current limitations are that it does not deal with the whole multi-field information, but aims every field separately, and to successfully apply the approach one needs to find representative isolevels sufficient for the comparison of the individual simulation time steps. Since this approach is the most general approach for our analysis task, we want to expand this approach by overcoming its limitations and propose a (multi-)field similarity-based approach.

There are many existing approaches for measuring similarity between spatial fields. An overview of the correlation measures for 3D fields is presented by Doncker [100]. Other measures use gradients instead of actual field values [101–103].

We also refer to Chapter 2 for an additional information about other multi-run multi-field simulation data analysis approaches.

8.3 Multi-run Similarity Plots

We consider the concept of multi-run similarity plots as a visualization approach for the analysis of ensembles of time-varying spatial simulation data. It uses MDS, where input information represents pairwise relations between all involved simulations’ time steps within an analyzed ensemble in the form of a distance matrix (DM). Projection based on the DM to a low-dimensional space results in visual representations of the relations in terms of strongest principal dimensions. Connecting points within the projected view, which are individual time steps, according to the temporal order forms polylines. Color coding them according to their simulation’s color, the resulting visualization is called multi-run similarity plot.

The main feature of the approach is that pairwise relations between simulations’ states, namely data field distributions over considered spatial domain, are to be defined. In the original version of the approach, isosurfaces were chosen as representative descriptors of field distributions. To compare them, a quasi-Monte Carlo approach is used to evaluate the scalar field at randomly chosen points and construct binary vectors rep-
resenting isosurfaces performing an inside-outside test. Based on these vectors, a distance function between the respective isosurfaces can be defined. The distance $d(A, B)$ between isocontours $A$ and $B$ is defined by the Jaccard distance [104]:

$$d(A, B) = 1 - \frac{M_{A \land B}}{M_{A \lor B}}$$  \hspace{1cm} (8.1)

Here $M_{A \land B}$ is the number of points inside both isocontours (logical and), and $M_{A \lor B}$ is the number of points inside of, at least, one of the isocontours (logical or).

The main advantage of the approach is its general applicability. The method can be efficiently applied to data of any spatial configuration (gridded or unstructured) and independently of the application area. Its visualization is flexible, that one can use projections to different visual spaces. In fact, the original paper used by default 2D and 3D projections including interactive switching between them and exchanging dimensions. We propose to first analyze the MDS eigenspace to determine the intrinsic dimensionality. Interestingly, for all data sets and all similarity measures used in this paper, the intrinsic dimensionality of the data space is 1, 2, or 3. In case of 1D, we propose to plot the changes within that 1D projected space over time. Interactive analysis with the projection allows to investigate its structure including linked views to spatial domain visualizations.

However, using isosurfaces as descriptors of the field distribution has several shortcomings. Most strikingly, one has to define an appropriate isovalue. A single isovalue is typically not enough for capturing an entire field. Even when identifying multiple representative isvalues for one field, there is no guarantee that they will remain representative for other time steps. Moreover different simulation runs behave differently, i.e., even when having identified a representative set of isovalue for all time steps of one run, it may still not be suitable for other runs. Another aspect of the existing approach is, that it considers only a single scalar field and is not dealing with multi-field data.
Hence, the contributions of our approach are mainly to overcome the listed shortcomings by:

- Extending the work for multi-run spatio-temporal data analysis using field similarity.
- Providing a new field similarity measure for the comparison of spatial field distributions, taking into account field information over the entire simulation domain.
- Discussing the effectiveness and applicability of the new field similarity measure in comparison to other field similarity measures.
- Extending the field similarity measure to a multi-field similarity measure and perform respective analyses.

8.4 Field Similarity

There are plenty of measures for the evaluation of similarity between sets of values. They come from different application areas such as geometry (Euclidean distance), statistics (correlation coefficient), data analysis (gradient-based measure), etc. Application areas define properties of the measures, which are aiming to capture certain characteristics of relations between compared objects. Our aim is a comparative analysis of (multi-)field data, i.e., considering the entire information of one time step of a simulation and comparing it to another. The measure should consider the spatial distribution within the field. Isosurface similarity does capture spatial distribution such as shape, but it does not consider all information available. We will first generalize the isosurface similarity measure to a field similarity measure, before we discuss other existing approaches against which we will compare our new measure afterwards.
Figure 8.1: Distribution of randomly placed points over a spatial domain are shown in yellow (enumerated). Three isocountours are shown with different colors. The corresponding binary vectors represent results of the inside-outside test.

### 8.4.1 Generalization of Isosurface Similarity Measure

Using isosurfaces as a descriptors of spatial field distribution considers only the chosen isolevel. To increase a sensitivity of the measure, one could consider more than one isolevel at once to form the descriptive vector by placing information about inside-outside test of the randomly distributed points for every isolevel next to each other instead of using them separately, see Figure 8.1. The impact of the individual isolevels decreases, but the whole descriptive vector contains information about all of them in an equal rate. Theoretically, there is no limitation for the number of participating isolevels, however the dimensionality of the descriptive vector increases by a factor of $l$ when considering $l$ isolevels. Obviously, for every spatial point there exists a single isovalue, where the inside-outside test switches from ”one” to ”zero”. Thus, that it is very easy to compute the $M_{A \& B}$ and $M_{A \lor B}$ from Equation 8.1 by counting how many isolevels cover every spatial point from the descriptive vector. Then, we refer to a field value at every point instead of computing and storing information about inside-outside tests for every
146

8.4. FIELD SIMILARITY

Figure 8.2: Illustration of intersection and union of isolevels in the limit, i.e., when assuming infinitely many isolevels.

isolevel. The result of such a comparison depends only on the number of isolevels (if the number is big enough we can consider them sampling the field range equidistantly) and field values at the spatial points. Using more isolevels increases the resolution of the comparison. As a limit case, the resolution will cover the whole field range. Thus, we can consider the actual field values at spatial points without sampling to isolevels. Assuming that the field range is normalized to the unit interval \([0, 1]\), we can generalize Equation 8.1 to

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

where \(A\) and \(B\) are the descriptive vectors, \(N\) is the number of randomly distributed points, \(\max()\) and \(\min()\) return biggest and smallest values out of the input pair. Whenever \(A = B\), we set \(D(A, B) = 0\). Figure 8.2 illustrates the measure using the idea of intersection and union as for isosurfaces. We also document the convergence of the isosurface similarity to our field similarity in the limit empirically in Section 8.5.4.

Using normalized field ranges, we can easily add information about all the data fields in one descriptive vector and perform a multi-field comparison using the same measure. It will increase the vector size by the number of considered fields, i.e., the complexity increases linearly with the number of fields.

The quasi-Monte Carlo approach uses \(N\) randomly distributed spatial samples. When considering gridded data, one could also take all the gridpoints. It would lead to a precise result, but the computation time would be high. In Chapter 6 we stated that a much smaller number suffices for very good approximations of their isosurface similarity. The same holds true for our generalized field similarity. For particle simulations, the particles change their positions over time such that the random sampling is required.
8.4.2 Pearson’s Correlation Measure

One of the well-known and widely used approaches for the estimation of similarity between distributions of values is calculating the correlation between them. A large number of correlation coefficients measuring this similarity exist. The most popular one is probably the Pearson correlation coefficient for measuring a linear correlation between two random variables:

The correlation coefficient between two sets $X$ and $Y$ is defined as follows:

$$r_{XY} = \frac{\sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{N} (y_i - \bar{y})^2}},$$

(8.3)

where $N$ is a set size.

Since the range of $r_{XY}$ is $[-1.0; 1.0]$, we want to shift and normalized it to the unit interval $[0, 1]$ by

$$r_{\vec{x}\vec{y}} = 0.5 \left( 1 - \frac{(\vec{x} - x_{mean})(\vec{y} - y_{mean})}{\| \vec{x} - x_{mean} \| \| \vec{y} - y_{mean} \|} \right),$$

(8.4)

such that $r_{\vec{x}\vec{y}}$ becomes a distance measure, i.e., it is equal to 0 in case of (full) positive correlation and equal to 1 in case of (full) negative correlation.

8.4.3 Average Gradient Measure

Considering spatial data fields, a common approach to study relations between them is to use a gradient-based comparison measures [101–103]. Computing gradients in every data point one can capture similarities in terms of dynamics instead of comparing absolute values. A common way to measure a field similarity using information collected by computing gradient-based relations for individual points is to find an average value of them over considered area of the domain. Computing advanced gradient-based measures (such as [102]) requires quite some computations to derive the distance matrix. Therefore, we decided to use a simple comparison of gradient vectors at individual points (chosen using the quasi-Monte Carlo approach) by computing the length of their differences. Then we define a similarity between two data fields as a half of average
difference length between the gradient vectors:

$$d_{X,Y} = \frac{\sum_{i=1}^{n} \| \vec{x}_i - \vec{y}_i \|}{2n}$$

(8.5)

Note that the distances $d_{X,Y}$ vary within the range $[0, 1]$ due to normalization.

8.5 Results and Discussion

Since we are using the multi-run similarity plot visualization, we decided to test the proposed measures on data sets described in Chapter 6 to compare with the results of using the isosurface similarity measure.

8.5.1 Application to Synthetic Data

First, a synthetic data set is used that represents three simulation runs, each of 100 time steps, describing the dynamics of Gaussian kernels, see Figure 8.3. Starting from one Gaussian kernel located in the middle of the 2D domain, the kernel splits while rotating around the center of mass with a constant angular speed and increasing distance to the center. In the simulation illustrated by the green arrows the kernel immediately splits into four kernels, while in the two other simulations it only splits into two kernels. While the simulation illustrated by the red arrows remain two, the simulation illustrated by the blue arrows split further into four kernels and ends up in the same final state as the simulation with green arrows.

Applying the concept of isosurface similarity to produce the multi-run similarity plot with a representative isovalue equal to 0.01, the resulting 2D projection exhibits a clear structure with a pendular pattern according to the rotational movement around the center of mass, see Figure 8.4 (a). Also, the red and blue separate in the middle of the simulation process. However, the rotational pattern of the green simulation run is not reflected in the first two principal dimensions of the projection. The ends of simulations are represented as expected: the green and blue simulation end up in the same place, while the red one ends up in a different place.
Figure 8.3: Schematic illustration of the synthetic data set introduced in Chapter 6 (Figure 6.2). For details we refer to Section 6.4. Three simulation runs represent the dynamics of Gaussian kernels over a 2D domain according to colored arrows.

(a) Isosurface Similarity  
(b) Generalized Field Similarity

(c) Average Gradient Similarity  
(d) Correlation Similarity

Figure 8.4: 2D multi-run similarity plots of three synthetic simulations computed using different similarity measures.
The projections computed using the proposed similarity measures are shown in Figures 8.4 (b-d). We observe that our generalization of the isosurface similarity measure in (b) results in a very similar projection. This was to be expected as the fields are very simple and a single isosurface actually captures the field quite well. The only difference is that now the green simulation run exhibits a slightly more complex pattern. The projections produced using gradient based measure (c) and the correlation measure (d) show somewhat different structures. They both exhibit some wavy patterns of all the simulation runs, but due to high curvature of the polylines it not straightforward to recognize them as the rotational movement of the kernels. Other features are present though. We conclude that all measures are doing quite well on this synthetic example.

### 8.5.2 Application to Astrophysical SPH Simulation Data

To test the proposed measures for computing multi-run similarity plots on a real-world data set, we chose the astrophysical simulation ensemble modeling a White Dwarfs system. These simulations are based on smoothed particles hydrodynamics, which means that the spatial domain is formed by a number of particles that change their positions and attributes over time. There are 45 simulation runs in the ensemble with two initial parameters varying, which are masses of the two stars. They are rotating around their center of mass until they merge. The result of this merging and the merging process itself are of interest and to be investigated in details considering the internal energy data field. In Figure 8.5 (a) when using isosurface similarity, one can easily distinguish three main simulation phases: the pendular pattern in the beginning of the simulations to the left, then a quick transition phase in the middle, and finally the dense and cluttered structure to the right. Investigating these structures using linked views to the domain visualizations one can conclude that they correspond to a stable rotational behavior, a short merging phase, and a final phase where the stars are merged.

However, this analysis was performed based on the information derived by an isosurface similarity measure with a certain isovalue equal to 0.29. Despite of the stable shape of the considered isosurfaces, there can be big differences in distribution of the internal energy. Applying the proposed similarity measures, the multi-run similarity plots are shown in Figures 8.5 (b-d). We observe that the first two principle dimensions of the
Figure 8.5: 2D multi-run similarity plots of 45 astrophysical simulations computed using different similarity measures. In (b) four points are selected corresponding to the last time steps of the simulations with index 8, 22, 34, and 44 for further analysis.

Our proposed generalized isosurface similarity measure does not capture the rotational behavior as one of the main data features, see Figure 8.5 (b). Instead, one can clearly see a diverging pattern with a few simulations behaving much differently from the others. Hence, there is a disagreement in the plots. While the isosurface similarity measure indicated similar values towards the end of the simulations, our field similarity measure indicates the opposite. This requires a further analysis, see below. We selected four simulations and picked their last time steps to investigate the reasons, why they were placed so far from each other. The selections are marked by yellow circles in Figure 8.5 (b) and with the corresponding simulation index. We rendered the corresponding representations of the spatial distribution of the particles, see Figure 8.6. Obviously, in
8.5. RESULTS AND DISCUSSION

(a) Simulation 8  
(b) Simulation 22  
(c) Simulation 34  
(d) Simulation 44

Figure 8.6: Visualizations of the spatial domain of the particle simulation for the data frames according to the selections in Figure 8.5 (b).

In the case of simulations 8 and 22, the merging process is still going. That is why they are clear outliers in the projection. The simulations 34 and 44 were placed closer to each other and they look similar in terms of spatial distribution. Note that the simulations 8 and 44 are very similar in the first principal dimension, but have a large difference in the second one. Despite of the similar spatial structure, the simulations 8 and 22 are placed far away from each other. To investigate this further, we built histograms of frequency of occurrences of the internal energy values for all the selected simulation steps, see Figure 8.7. Now it becomes visible, that simulation 8 contains a huge number of points with a low internal energy, which was captured by our measure.

The gradient-based similarity captures the initial rotation as one of the main principal dimensions of the projection, see Figure 8.5 (c). Except of the rotational pattern we can see how the simulations develop when merging and again there is no stable final simulation phase observed. However, the differences between the simulation runs are not so strong, as in Figure 8.5 (b), but we can still identify a couple of outliers.
8. COMPARATIVE VISUALIZATION OF MULTI-RUN MULTI-FIELD TIME-VARYING SPATIAL DATA

Figure 8.7: Histograms representing frequency of occurrence of the internal energy field values for the data frames according to the selections in Figure 8.5 (b).

The correlation-based similarity captures the rotational pattern as a main data feature, and therefore we can see a very clear circular structure on the projection to the first two principal dimensions, see Figure 8.5 (d). One can again distinguish the three simulation phases: the outer circle, a less dense transition phase, and the center. This measure did not capture differences towards the end of the simulations.

It is of interest to investigate the projection space for other principal dimensions. Magnitudes of their eigenvalues allow us to estimate usefulness of the corresponding dimensions in terms of the visual analysis (indicating inherent dimensionality). We plot the magnitudes of the first 16 principal dimensions for the projections derived by the proposed field similarity measures, see Figure 8.8. It is clearly visible, that for all of the projections only the first principal components are valuable, while others are negligibly small. In case of Figure 8.5 (d) the first two eigenvalues are of the same order, and as we mentioned above they represent the rotational movement of the stars. In two other cases (see Figure 8.5 (b and c)) we observe that magnitudes of the first eigenvalues are much higher then for all the rest. In this case it makes sense to plot the corresponding first principal component over time, as shown in the Figure 8.9.
8.5. RESULTS AND DISCUSSION

Figure 8.8: Magnitudes of the eigenvalues for the first 16 principal dimensions of the projections shown in Figure 8.5 (b-d). The red line corresponds to the projection using our field similarity (b), the green one to the gradient-based similarity (c), and the blue one to the correlation-based similarity (d).

Figure 8.9: Multi-run similarity plots of the first principal dimension of the projection over time for the astrophysical simulations.

The projections plotted over time are less cluttered. To relate the simulations to their parameter space, we color-code the simulation runs by the two simulation parameters, namely the sum of the star masses and the difference of the star masses. We use a 2D color map, where the sum is indicated by changing hue from green to red and the difference is indicated by changing the saturation from high to low. We can easily distinguish the dynamics of the simulations’ changes over time in a comparative manner. We can
observe in Figure 8.9 (a) that the pendular pattern, which represents the rotational behavior, appears only for the outliers. As we can see, the main feature captured by the first principal component is the final simulation state, which exhibits a big dispersion for the considered simulations. In Figure 8.9 (b), we observe a similar structure, but in this case the first principle component contains information about both rotational behavior and dispersions. We conclude that depending on the analysis purposes, each of the projections can be desired. If we consider the rotational pattern as noise hiding the main trend, then our generalized measure is preferable.

To evaluate the impact of the initial parameters, we filtered the runs with desired parameter values. In Figure 8.9 (c) we show only simulations with equal star masses and their sum increases from the lowest to the highest value. We observe a straight dependence between the simulations’ results and this parameter reflected with an certain order. In Figure 8.9 (d), we show runs with varying mass difference (keeping the mass of one star fixed and large). We again observe a straight dependence between the simulation results and the mass difference.

8.5.3 Application to Multi-field Climate Simulation Data

The concept of application of the proposed measures to a multi-field data does not differ from computing similarities between single field data frames. As was described in Section 8.4, we expand the descriptive vectors by adding information about all the analyzed data fields (normalized in advance). We have already shown, that the proposed measures capture data features successfully. Depending on the purposes of the analysis one can choose the desired measure to compute the multi-run similarity plot. Now we want to show, that it is possible to consider several data fields simultaneously and estimate the simulations’ deviations in terms of the entire data information.

The data set we analyze is an ensemble of 11 simulations of an ocean dynamics world model. The main attributes of the data are ice thickness, ice concentration, and snow thickness. A duration of the simulations is one year and they differ by coupled integrations during the simulation time. Our goal is to estimate a spreading of the simulations during the simulation time and identify outliers.
First, we compute similarity plots for the three data fields separately using our generalized isosurface similarity measure, see Figure 8.10 (a-c). All the computed projections have the first principal dimension much stronger (i.e. high eigenvalue) than all the others. Hence, it make sense to plot the information from the main principle dimension over time. In the plots, we can clearly see the seasonal structure: starting in January, the amount of ice and snow is increasing until the spring/summer season, where it decreases until the fall/winter season. The different fields exhibit different structures, but the highest deviations can be observed at the peaks of the main seasons. In Figure 8.10 (a) one can identify the magenta outlier, while in Figure 8.10 (b,c) there are just few deviations of all the simulation runs during different time intervals.

Figure 8.10: (a)-(d) Multi-run similarity plots of ocean dynamics world model. The horizontal axis represents time, while the vertical axis represents the first principal dimension of the projection. (e) Plot of deviation from mean values for every simulations’ time step.
Computing the multi-run multi-field similarity map for these three data fields, we obtain a structure that looks like an average out of all the previous plots, see Figure 8.10 (d). Note that the difference between an averaging and computing a projection considering all three data fields is that for every projection there are own principal dimensions such that an averaging would actually not be correct. The result still allows us to identify the outliers from the single field projections, but they are weaker, because the seasonal pattern is very strong and does not allow for resolving the deviations in the behavior.

To overcome this issue we propose to plot only the deviations from the mean of every simulations’ time step, see Figure 8.10 (e). This removes the seasonal pattern and we can identify outliers, which are the magenta, purple, and yellow simulation runs. These simulations can be identified as outliers in (a) and (b) correspondingly. However, the blue simulation, which is deviating a lot in Figure 8.10 (c) is not an outlier in the multi-field projection. Its outstanding impact to a single data field was not supported by other fields and this simulation is not an outlier in terms of the main principle dimension derived by the multi-run measure.

8.5.4 Discussion

We showed above that the proposed field similarity measures are suitable for comparing spatial data fields in application to comparative visualization of multi-run multi-field simulation data using the multi-run similarity plot approach. However, to use them properly a qualitative comparison of the measures is desirable.

In order to compare the measures we consider three pairs of structurally different 2D scalar fields and compute similarities between of them. The domain is defined in $\mathbb{R}^2$ by $\{x \in [-1, 1], y \in [-1, 1]\}$. In Figure 8.11 (a), the field distributions are defined by the rules $\sqrt{(x - 0.25)^2 + y^2}$ and $\sqrt{(x + 0.25)^2 + y^2}$. In Figure 8.11 (b) the field distributions are defined by the rules $\sqrt{x^2 + y^2}$ and $-\sqrt{x^2 + y^2}$. In Figure 8.11 (c) the field distributions are defined by the rules $\sqrt{x^2 + y^2}$ and $x^2 + y^2$. The results of the computation of similarities using the different field measures are listed in the Table 8.1.
8.5. RESULTS AND DISCUSSION

Figure 8.11: Three pairs of 2D scalar fields generated with corresponding rules. The color map is the same as in Figure 8.3.

Table 8.1: Values of dissimilarity between the 2D data fields shown in Figure 8.11 computed using the proposed similarity measures and others and corresponding timings. Note that in all measures 0 means complete similarity and 1 means a complete dissimilarity.

<table>
<thead>
<tr>
<th>Similarity Measure</th>
<th>Figure 8 (a)</th>
<th>Figure 8 (b)</th>
<th>Figure 8 (c)</th>
<th>time, ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generalized isosurface similarity measure</td>
<td>0.31947</td>
<td>0.69864</td>
<td>0.11860</td>
<td>0.627</td>
</tr>
<tr>
<td>Average gradient</td>
<td>0.00130</td>
<td>0.00276</td>
<td>0.00127</td>
<td>1.213</td>
</tr>
<tr>
<td>Pearson's coefficient</td>
<td>0.01056</td>
<td>1.0</td>
<td>0.01272</td>
<td>1.319</td>
</tr>
<tr>
<td>Gradient based measure</td>
<td>0.00130</td>
<td>0.01104</td>
<td>0.01446</td>
<td>2587</td>
</tr>
<tr>
<td>Spearman's coefficient</td>
<td>0.30203</td>
<td>1.0</td>
<td>0.0</td>
<td>389</td>
</tr>
</tbody>
</table>
The resulting similarities are very different. Note that the values obtained from different measures are not comparable with each other. Despite of the same range of possible distance values (which is \([0, 1]\)), their interpretations are quite different. The introduced generalization of the isosurface similarity measure takes into account similarities between all layers of the existing field values and returns 0 only in case of an absolute match between the fields, while value 1 is only possible in case of two constant fields of values 1 and 0. Also, as shown in the table, this measure is the fastest one.

The proposed average gradient measure returns an average value of a length of all the computed differences between gradients in spatial points. It means that value 0 appears in the case when fields match up to a constant difference. The value 1 is possible when all the differences between considered gradients are equal to 2 (i.e., the gradients have maximum length equal to 1.0, but inversely oriented), which is impossible on 2D data fields. Due to the averaging the resulting value is likely to be rather low, but this does not mean that the computed values in the distance matrix are not representative. The MDS projection approach places the points based on relations between of them, therefore despite of low actual values in the distance matrix, the structure of the projection is still representative.

The proposed Pearson’s correlation measure results in a linear correlation between the sets of values, which are coordinates of the descriptive vectors. The similarity value 0 is possible in case fields match up to a constant difference. The value 1 appears in the case of an inverse match up to a constant difference.

We also investigate another gradient-based approach [102]. Averaging the respective measure, we achieve similarly distributed results to the average gradient measure. The Spearman’s correlation coefficient can also be used for comparing the data fields. This measure is rank-based and provides different results capturing data features quite differently. However, these two measures are much more complex than the proposed measures and inefficient in terms of computation time for our purposes.

Overall, we can conclude that our new field similarity measure is quite different from gradient- and correlation-based approaches.
Finally, we also want to show empirically that our new measure is indeed a generalization of the isosurface similarity and we want to observe the convergence. Thus, we compute the isosurface similarity measure by taking more and more samples. Figure 8.12 shows how the difference between the isosurface similarity to our field similarity decreases with increasing number of isolevels and that they converge. This confirms the high effectiveness of our proposed measure to capture field distributions. However, our approach is much more efficient, as a combination of isosurfaces would require more than 100 isolevels to be considered for a similar accuracy.
8.6 Conclusion

We have presented a new approach to compute similarities between data fields and applied it within a framework for comparative visualization of multi-run multi-field time-varying spatial data. We extended an existing technique for computing multi-run similarity plots by replacing the concept of using isosurfaces as a data field descriptor with using the entire field information. We derived our field measure as a generalization of the isosurface measure. We showed that also other field similarity measures can be successfully deployed depending on different purposes of data analysis. We compared them by applying them to different synthetic and real-world data sets. The new concept allows for a faster analysis without any knowledge about data underlying, such as a representative isovalue. An opportunity of using different measures for computing multi-run similarity plots allows to make a comprehensive overview of a simulations’ ensemble and in combination with visual analysis tools helps to investigate desired data features and to search for unknown data features. The multi-run similarity plot approach is independent of spatial data structures and is flexible in choosing a suitable similarity measure. Hence, it could even be applied to categorical or non-spatial data.
Chapter 9

Conclusions and outlook

In this work we have proposed a number of approaches and tools for a visual analysis of multi-run multi-field spatio-temporal simulation data. They were developed and designed to perform an efficient and interactive visual analysis on every level of the considered task: from a very first overview of entire ensembles down to comparative and qualitative analysis of individual simulation time steps. All of the proposed methods are applicable to data of any spatial configuration (e.g., gridded or point based spatial domains).

The proposed visual analysis pipeline consists of several data visualizations depending on considered data aggregation levels. On the highest level we consider information about the entire ensemble using field distribution histograms. They show distributions of occurrences of field values over all simulations and their time steps. Using interactive tools one can investigate the histograms and make first conclusions to narrow down an area for further analysis (e.g., define a field range of interest or identify simulations with outstanding values).

The proposed function plots allow to distinguish behavior patterns of individual simulations, as well as to perform comparative analysis of multiple simulations or between different data fields. Aggregating frequencies of occurrences of field values by time steps, we plot function values for every spatial point over an entire simulation duration. Applying a transfer function with an adjustable color scheme to the plot, we highlight
and investigate data features. Introduced interactions with the plots allow to make selections on the plot with a further progressive rendering of function trajectories for the selected points for different data fields, as well as render them in physical domain. Also such visual representation of simulations allows to align them in time by a desired feature, and, as confirmed by our example, even do the alignment automatically. This step of the pipeline allows to significantly increase an accuracy and efficiency of other techniques for a further data analysis by precise definition of area of interest and detection of general data features.

For a comparative visual analysis of multi-run multi-field simulation data, we proposed similarity plots, which encode each simulation as a polyline of connected points (simulation’s time steps) in a low dimensional visual output, projected from a similarity matrix. Distances between the points on the plot represent dissimilarities between corresponding data frames. The proposed similarity measure allows to compare data frames using information about a single selected scalar field, multiple fields or the entire multi-variate structure. Using numerous interaction tools one can easily investigate the projection, e.g., by filtering, color coding, selections, etc. Also it is possible to consider different dimensions of the projection and represent them in combination with each other, as well as according to a time line. Linked views to physical domain and parameter space help to understand identified features and data structure. We successfully applied proposed visualizations (field distribution histograms, function plots and similarity plots) to real world data sets and approved them with the corresponding domain experts.

For a detailed analysis of scalar fields, we developed efficient algorithms for rendering of continuous scatterplots. They allow to investigate the structure of relations between different continuous data fields. Our algorithms make a rendering of such plots possible for data of any spatial configuration and in interactive fashion using GPU parallelization and a progressive rendering technique.

To make a comparative analysis of volumetric data and to compute the similarity matrices, we introduced novel similarity measures. They allow to compare individual data frames independently of spatial configurations for a selected iso-value, as well as consider data frames in general with a single scalar field or multi-field structure. We
showed that the proposed isocontour based similarity measure can be successfully and with a high efficiency used for investigation of a structure of a single data frame using isosurface similarity maps. We compared our proposed measures with other commonly used existing measures and confirmed their usefulness by applying to different data.

The proposed methods and tools for visual analysis of multi-run multi-field spatio-temporal simulation data provide an opportunity to perform the analysis on every scale from considering whole simulation ensembles down to an individual data frame. They can be used stand-alone or in combination with other existing tools. Their effectiveness was approved by numerous examples on real world data from different application areas, feedback from the domain experts and user study.

Based on the results of this work many research directions and application scenarios can be investigated further. For example, the concept of multi-run similarity plots can be potentially applied to non-spatial data or even to categorical data. Developing of proper similarity measures for such data is a challenge. Analysis of the parameter space together with the projection space helps to understand sensitivity of the simulation model and its reaction, as well as to estimate parameters’ influence. This information can be potentially used to predict simulation’s behavior and to guide researchers for their further simulations experiments to save their time and increase their efficiency. Flexibility of the presented approaches allows to adjust them and to focus on any particular application task if necessary by modifying of the parameters of a considered simulation domain, similarity measure, MDS projection approach, or visualization tools.
Author’s publications


Bibliography


[72] A. Bordignon, R. Castro, H. Lopes, T. Lewiner, and G. Tavares, “Exploratory visualization based on multidimensional transfer functions and star coordinates,”


