FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Scalable Comparative Visualization

Visual Analysis of Local Features in Different Dataset Ensembles

DISSERTATION

zur Erlangung des akademischen Grades

Doktorin der Technischen Wissenschaften

eingereicht von

Dipl.-Ing. Johanna Schmidt
Matrikelnummer 0025558

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller

Diese Dissertation haben begutachtet:

Eduard Gröller	Stefan Bruckner	Timo Ropinski

Wien, 17. Mai 2016

Johanna Schmidt

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Scalable Comparative Visualization

Visual Analysis of Local Features in Different Dataset Ensembles

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktorin der Technischen Wissenschaften

by

Dipl.-Ing. Johanna Schmidt
Registration Number 0025558

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller

The dissertation has been reviewed by:

Eduard Gröller	Stefan Bruckner	Timo Ropinski

Vienna, 17th May, 2016

Johanna Schmidt

Erklärung zur Verfassung der Arbeit

Dipl.-Ing. Johanna Schmidt
Krongasse 20/11, 1050 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 17. Mai 2016

Johanna Schmidt

Acknowledgements

I would like to express my very great appreciation to my supervirsor Meister Eduard Gröller, for his valuable and constructive suggestions during this research work. Thank you for giving me the chance to work on this research project. I am also particularly grateful for the assistance and advices given by Stefan Bruckner. The success of this research was also greatly supported by the positive working atmosphere at the institute, and for this I would like to express my thanks to Artem Amirkhanov, Michael Birsak, Michael Hecher, Alexey Karimov, Peter Mindek, Gabriel Mistelbauer, Reinhold Preiner, Bernhard Steiner, Johannes Sorger, Viktor Vad, Ivan Viola, Manuela Waldner and Nicholas Waldin. My special thanks go to Johannes Kehrer for proof-reading the introduction of this thesis. I would like to thank the technicians Stephan Bösch-Plepelits, Simone Risslegger and Andreas Weiner for their great technical support during the thesis. I wish to especially acknowledge the help provided by Anita Mayerhofer, without whom the organisation of the institute would me much more complicated.

The research work carried out during this thesis has been partially supported by the ViMaL project (FWF - Austrian Research Fund, no. P21695) and by the AKTION project (Aktion OE/CZ grant number 64p11).

None of this would have been possible without the patience and support of many people. Special thanks to mum and dad, for encouraging me to follow this path in my life. Many thanks to my brother and to my sisters, for their support as well as for distracting me from work when it was necessary. The person who was probably affected the most from this thesis (apart from me) was Christian, so many thanks for accompanied me in completing this project, and thank you for your patience during the times of irregular working hours. Thanks to my friends, who helped me stay sane through these difficult years. Thanks to Apollonia, for your distraction because you needed to got for a walk, which that actually lead to the creation of interesting new ideas.

Nothing in this world, that's
worth having comes easy
Dr. Kelso, Scrubs

Kurzfassung

Die Möglichkeit, Daten schnell, interaktiv und visuell vergleichen zu können, wird für die Datenanalyse eine immer wichtigere Aufgabe. In der Visualisierung sind immer mehr Systeme gefragt, die sich nicht nur für die Repräsentation von einzelnen Datensätzen, sondern für die Analyse von ganzen Sequenzen von Datensätzen eignen. Der Benutzer kann beim Vergleichen von Daten auf zwei Arten unterstützt werden. Zuvorderst ist es sehr hilfreich, wenn Benutzer die Datensätze, die verglichen werden sollen, im System passend zueinander anordnen können. Das unterstützt die intuitive Vorgangsweise von Menschen beim Vergleichen von Daten. Weiters können Visualisierungssysteme die Unterschiede in den Daten selbst berechnen und dann in geeigneter Form dem Benutzer präsentieren. Die *Vergleichende Visualisierung* beschäftigt sich mit neuen Techniken, wie man in der Visualisierung Benutzer am besten beim Vergleichen von Daten unterstützen kann. Solche Techniken können üblicherweise einfach für zwei oder mehrere Objekte angewendet werden, stoßen aber an ihre Grenzen, sobald die Datenmenge entsprechend groß wird (z.B. 100 Objekte oder mehr). Solche Datensammlungen, die eine große Anzahl an individuellen, aber doch zusammengehörigen, Datensätzen enthalten, werden *Ensembles* genannt. Die einzelnen Datensätze, genannt die *Ensemble-Mitglieder*, beschreiben dabei dasselbe Phänomen, weisen aber kleine lokale Unterschiede auf. Ursprünglich stammen Ensembles aus dem Bereich der Simulationsanalyse, meist für Wetter- und Klimadaten. In diesen Bereichen werden sie schon seit einiger Zeit verwendet, da mehrere Simulationsläufe immer zu einer großen Anzahl von Resultaten führen, die anschließend analysiert werden müssen. Die Simulationsanalyse war daher ein starker treibender Faktor im Bereich der *Ensemble-Visualisierung*. Leistbare Rechenkapazitäten und die Verfügbarkeit von unterschiedlichsten Analysealgorithmen (z.B. für die Segmentierung) haben aber dazu geführt, dass sich auch andere Anwendungsbereiche heutzutage mit der Analyse von Ensembles beschäftigen müssen. Ensembles werden üblicherweise entweder basierend auf Datenmerkmalen (*feature-based*), oder basierend auf lokalen räumlichen Regionen (*location-based*) analysiert. Im Falle der *Analyse basirend auf lokalen räumlichen Regionen* müssen Visualisierungssysteme Möglichkeiten anbieten, dass Benutzer ihre Analyse auf lokale Regionen konzentrieren können.

In Rahmen dieser Arbeit wurden verschiedene Techniken für das visuelle Vergleichen von komplexen Daten entwickelt. Ein spezielles Augenmerk wurde dabei auf die Skalierbarkeit der Techniken gelegt, und zwar im Bezug auf die mögliche Anzahl von Mitgliedern pro Datensatz. Die Techniken operieren auf unterschiedlichen Arten von Datensätzen in 2D und 3D. Im ersten Teil dieser Arbeit wird eine Technik für die Analyse von 2D Bilddaten vorgestellt, die nicht nur die Berechnung von lokalen Unterschieden in den Daten ermöglicht, sondern auch eine genauere

Einsicht in die Daten erlaubt. Dadurch kann, im Unterschied zu bestehenden Methoden, sehr schnell festgestellt werden, wo sich die Daten unterscheiden, und auf welchen Merkmalen diese Unterschiede beruhen. Dadurch werden Muster in den Daten sichtbar, und es können sehr schnell Sonderfälle lokalisiert werden. Der zweite Teil der Arbeit befasst sich mit einem System, das die Analyse von einem Ensemble bestehend aus dreidimensionalen Objekten (*meshes*) ermöglicht. Solche Ensembles werden beispielsweise beim Testen von Rekonstruktionsalgorithmen für Punktwolken mit unterschiedlichen Parametern generiert. Ähnlich wie die vorgestellte Technik zum Vergleichen von 2D Bilddaten kann das System auf eine große Anzahl an Datensätzen angewendet werden und ermöglicht sowohl die Berechnung der Unterschiede, als auch die lokale Analyse von einzelnen Regionen in den Daten. Die lokale Analyse erfolgt in diesem Fall im 3D, da es sich um 3D-Datensätze handelt. Das vorgestellte System bietet auch die Möglichkeit, lokale Unterschiede in den Daten mittels paralleler Koordinaten zu visualisieren. Vorher selektierte und vom Benutzer selbst gewählte Regionen dienen dabei als Koordinatenachsen, und die 3D-Datensätze (*meshes*) werden als Polylinien in den Plot eingetragen. Dadurch kann sehr schnell abgelesen werden, welche Datensätze in welchen Regionen gute/schlechte Ergebnisse liefern. Aufbauend auf dieser Idee wird im dritten und letzten Teil dieser Arbeit eine weitere 3D-Technik vorgestellt, die die Analyse von lokalen Regionen in einem Ensemble von Volumsdatensätzen ermöglicht. Benutzer können in diesem Fall lokale Regionen, die für die Analyse von Interess sind, selbst wählen. Basierend auf der Ähnlichkeit der Regionen, können diese in einem Graphen angeordnet werden. Durch den Graphen können Regionen mit einer ähnlichen Charakteristik im Ensemble gefunden werden, und einzelne Mitglieder gegen den Rest des Ensembles verglichen werden. Alle vorgestellten Techniken wurden auf aktuelle Datensätze aus verschiedenen Anwendungsgebieten angewandt, und die Resultate der Analyse belegen die Nützlichkeit der Techniken für die vergleichende Analyse von Ensembles.

Abstract

The comparison of two or more objects is getting an increasingly important task in data analysis. Visualization systems successively have to move from representing one phenomenon to allowing users to analyze several datasets at once. Visualization systems can support the users in several ways. Firstly, comparison tasks can be supported in a very intuitive way by allowing users to place objects that should be compared in an appropriate context. Secondly, visualization systems can explicitly compute differences among the datasets and present the results to the user. In *comparative visualization*, researchers are working on new approaches for computer-supported techniques that provide data comparison functionality. Techniques from this research field can be used to compare two objects with each other, but often reach their limits if a multitude of objects (i.e., 100 or more) have to be compared. Large data collections that contain a lot of individual, but related, datasets with slightly different characteristics can be called *ensembles*. The individual datasets being part of an ensemble are called the *ensemble members*. Ensembles have been created in the simulation domain, especially for weather and climate research, for already quite some time. These domains were greatly driving the development of *ensemble visualization* techniques. Due to the availability of affordable computing resources and the multitude of different analysis algorithms (e.g., for segmentation), other domains nowadays also face similar problems. All together, this shows a great need for ensemble visualization techniques in various domains. Ensembles can either be analyzed in a feature-based or in a location-based way. In the case of a *location-based analysis*, the ensemble members are compared based on certain spatial data positions of interest. For such an analysis, local selection and analysis techniques for ensembles are needed.

In the course of this thesis different visual analytics techniques for the comparative visualization of datasets have been researched. A special focus has been set on providing scalable techniques, which makes them also suitable for ensemble datasets. The proposed techniques operate on different dataset types in 2D and 3D. In the first part of the thesis, a visual analytics approach for the analysis of 2D image datasets is introduced. The technique analyzes localized differences in 2D images. The approach not only identifies differences in the data, but also provides a technique to quickly find out what the differences are, and judge upon the underlying data. This way patterns can be found in the data, and outliers can be identified very quickly. As a second part of the thesis, a scalable application for the comparison of several similar 3D mesh datasets is described. Such meshes may be, for example, created by point-cloud reconstruction algorithms, using different parameter settings. Similar to the proposed technique for the comparison of 2D images, this application is also scalable to a large number of individual datasets. The application enables the

automatic comparison of the meshes, searches interesting regions in the data, and allows users to also concentrate on local regions of interest. The analysis of the local regions is in this case done in 3D. The application provides the possibility to arrange local regions in a parallel coordinates plot. The regions are represented by the axes in the plot, and the input meshes are depicted as polylines. This way it can be very quickly spotted whether meshes produce good/bad results in a certain local region. In the third and last part of the thesis, a technique for the interactive analysis of local regions in a volume ensemble dataset is introduced. Users can pick regions of interest, and these regions can be arranged in a graph according to their similarity. The graph can then be used to detect similar regions with a similar data distribution within the ensemble, and to compare individual ensemble members against the rest of the ensemble. All proposed techniques and applications have been tested with real-world datasets from different domains. The results clearly show the usefulness of the techniques for the comparative analysis of ensembles.

Contents

CHAPTER 1

Introduction

When we **visually compare** objects, we try to find the differences and similarities by using our eyes. Since vision is our dominant sense, we are well-trained for visually putting objects into relation. In our early developments, we intuitively make use of visual comparison to understand the concepts of geometry and measurements. Later on, we are faced with visual comparisons in many situations in our everyday life. We use visual comparison to make simple decisions like selecting between two similar photographs, and we need comparisons to do more detailed and fine-grained analyses in our professional life. For people working, for example, in chemistry, medical treatment, cosmetics, or photography, visual comparisons are a necessary step to judge upon someone's work, to evaluate results, and to improve the current state of a certain product. Since visual comparison is a very intuitive and easy-to-understand concept, it is also used for entertainment, as for example in the popular *spot-the-difference* puzzles. There people should find

Figure 1.1: Visual comparison. Humans are very well trained to visually compare objects. The concept of visual comparison is also used for entertainment, like in the popular *spot-the-difference* puzzles with two images [Wik16].

the differences between two rather similar images, where the images have usually been altered with photo manipulation. Spot-the-difference puzzles are often used in activity books for children or newspapers. An example for such a puzzle can be seen in Figure 1.1.

This thesis deals with the **visual comparison of datasets** of different types and domains. A special focus lies on the scalability of techniques, meaning how visual comparison can be applied, if a large number of objects need to be compared with each other. The objects can be of different complexity, like data in 2D (e.g., images) or data in 3D (e.g., volumes). This scalability with aspect to the number of datasets, and by considering the complexity of the objects, poses several challenges, which are discussed in this thesis. Solutions are proposed for the comparison of 2D images, 3D meshes and volume datasets.

In the first part of this chapter, the general scope of the thesis topic and the necessary background knowledge is described (Section 1.1). In the second part, the related work relevant for this thesis is introduced (Section 1.2). In the third and final part of the chapter, the contributions of the thesis are outlined (Section 1.3).

1.1 Scope of the Thesis

Although humans often solely rely on their visual system to compare objects, it is also possible to actively **support comparison tasks**. A demonstration of the possible support mechanisms, as investigated by Tominsky et al. [TFJ12], can be found in Figure 1.2. A very intuitive way to aid the comparison, and this is what humans usually do, is to place the objects that should be compared next to each other. This makes it much easier to visually switch between the objects, and this way spot the differences. If the objects are spatially large, it may additionally be helpful to partially overlay them with their neighbors, so that the regions of interest are closer together. Another technique, also often used in our everyday life, is to place objects in a stacked way (i.e., above each other) and flip between them. *Flip-books* use this concept, where a series of gradually varying pictures are turned rapidly, so that the pictures appear to be animated. The same technique can be used to compare objects (by flipping forth and back). If it is possible to represent the objects that should be compared in a semi-transparent way, they could be placed above each other in front of a light source (e.g., a window), so that differences become visible due to the manually created *blending effect*. Another intuitive comparison method, especially if the results need to be remembered, or communicated to others, is the explicit marking of differences on the objects themselves (e.g., by encircling interesting features with a pencil).

Comparisons are not only an everyday's concept we have to deal with, they also play a more and more important role in **visual data exploration and analysis**. Datasets are getting more and more complex, and increasingly visualization systems need to allow the users to relate and compare parts of the data, or whole datasets. This is necessary so that experts can gain insight into the data, formulate, confirm, fine-tune, or reject initial hypotheses, and get a better understanding of the available data. Visualization systems can support comparison tasks in two ways. Firstly, visualization systems can allow the users to place objects in a way that supports the comparison task. This imitates the natural workflow users typically employ when comparing

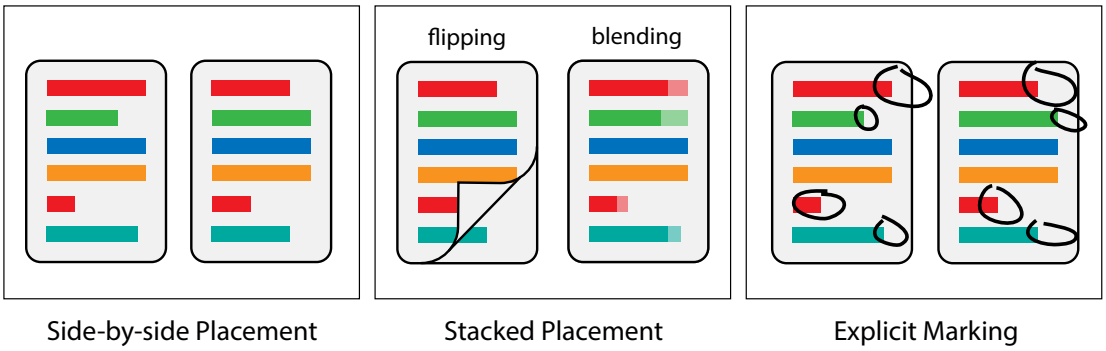

Figure 1.2: Supporting comparison tasks [TFJ12]. Humans intuitively arrange objects *side-by-side* if they want to compare them. Alternatively, if possible, objects can be placed in a *stacked* way, so that differences become visible by *flipping* the objects, or through *blending* (i.e., semi-transparency). If differences should be remembered, or communicated to others, they are often *explicitly marked* by directly painting onto the objects.

objects. Visualization systems following this concept typically allow the users to specify the objects of interest in the data (e.g., by applying selection or filtering interactions). Then the visualization system provide means so that users can arrange the objects of interest in a way that suits the comparison task. Possible modalities range from arranging objects in a grids, to the usage of multiple interactive views. An important task of the visualization system is to resolve occlusions that might occur if objects are, for example, placed above each other. Occlusions can be resolved by applying blending or folding effects. In all cases the task of finding and evaluating the differences is left to the users' perception. Therefore, secondly, visualization systems can also compute the differences among datasets and present them to the users. Computers are better in fine-grained calculations than human, and are therefore in some cases better suited to automatically detect differences in datasets. Using visualization system this way implies that a metric for comparing the objects, and a concept for presenting the differences to the users, exists beforehand. One drawback of this approach is that the more the visualization system itself is involved in the comparison workflow, the more the visualization system is typically targeted towards a specific comparison scenario.

In **comparative visualization**, researchers work on automated and semi-automated techniques that are designed to explicitly support comparison tasks of complex objects. Visualizations in this case do not only deal with tabular data, but also with more complex objects including graphs, surfaces, or volumes. Comparative visualization can be used in several different application domains [Hin09], ranging from medicine, to data mining, to material sciences. If visualization systems should be employed by users not familiar with computer science and automatic data analysis, the interactive concepts should be as easy to understand as possible. Therefore, comparative visualization techniques often make use of the intuitive concepts humans would use when comparing objects. Gleicher et al. [GAW+11] observed that the visual design strategies for comparative visualization can be divided into three categories: *juxtaposition* (placing objects next to each other), *superposition* (placing objects in the same coordinate space), and *explicit*

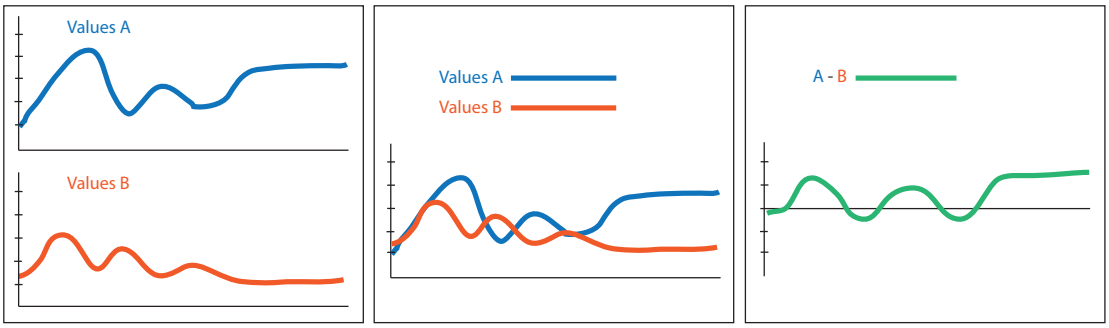

Figure 1.3: Visual designs for comparative visualization [GAW+11]. If *juxtaposition* is used, the objects that should be compared are placed next to each other. The design concept of *superposition* places objects in the same coordinate space. If *explicit encoding* is used, the differences among the objects are visually encoded.

encoding (visual encoding of the differences). An overview of the design strategies is given in Figure 1.3. It can be seen that these three categories actually match the intuitive concepts used by humans when comparing objects (as it has been illustrated in Figure 1.2). The design strategy of juxtaposition describes that objects that are compared are placed next to each other. In the case superposition is used, objects that should be compared are placed in the same coordinate space. Superpositions are typically used in situations with limited screen space, and where the objects to be compared are similar enough, so that they can be viewed in the same coordinate space. Both design strategies of juxtaposition and superposition rely on the users' perception to make connections between the objects, and to spot the differences. Explicit encoding means that differences between objects are computed beforehand, and are then explicitly visually presented to the users. This requires a pre-definition about the relationships between the objects, about the importance of the relationships, and a proper metric to extract this information. Explicit encoding, therefore, provides a trade-off. The users definitively save time during the analysis, since the differences are directly presented to them. On the other hand, explicit encoding is limited to datasets with existing pre-defined relationships. Explicit encoding is, therefore, mainly used in cases where the relationships between the objects are the actual topic of the analysis.

If visualization systems compute differences between the datasets themselves, then the systems are actively involved in the **datasets comparison**. For comparing datasets, Verma and Pang [VP04] introduced three modalities of how data can be compared. As an exemplary use case, they were working on the visual comparison of flow data. There they defined that datasets can be compared at three different levels: on an *image-based*, on a *data-based*, or on a *feature-based* level. The image-based comparison is considered to be the most simple one. In this case images (e.g., the output of visualization algorithms) are compared using standard image comparison methods [ZCW02]. The image-based comparison is useful to compare different representations of the same dataset (e.g., a volume rendered with different transfer functions). In the case of data-based comparison, the analysis concentrates on the raw data values of the datasets and compares

them. The advantage of this type of comparison is that differences in the data can be explicitly presented to the users. The greatest challenge, though, is the design of a suitable data comparison metric. This then also implies a disadvantage of this type of comparison, because the design of a metric is often task-dependent, which makes this concept less general. The feature-based comparison first extracts features from the data, and then compares the features in all datasets. Features may be application-dependent (e.g., shock waves, vortices) or application-independent (e.g., streamlines, iso-surfaces). The feature-based comparison allows users to concentrate on derived properties, that often have semantic meanings.

The need for automatic data comparison in visualization becomes even more obvious when the users are not only dealing with very complex, but also very **large datasets**. It is nowadays possible to create collections with a huge number of individual datasets (i.e., up to 100 items, or more). Comparative visualization techniques may be used for just two datasets as well, but their impact especially becomes obvious if they are applied to a large amount of data. Originally, large datasets have been used in the simulation domain for already quite some time [MGJ+10]. Especially the domain of weather and climate research was a driving force for the analysis of large datasets [NFB07]. The used prediction models comprise a large variety of parameter values, and, in addition, usually do not lead to one fixed result, but to a spectrum of possible outcomes. An example for such an ensemble dataset can be seen in Figure 1.4. It is a special challenge in visualization to find out how comparisons can be done at such a large scale.

Originating from the simulation domain, such large data collections (that also exhibit certain characteristics) are called **ensembles**. The individual datasets being part of an ensemble are called the **ensemble members**. The definition of ensembles is on the one hand strongly targeted to simulation results, but on the other hand may be also applied to other data collections. Based on the definition by Wilson et al. [WP09], we define ensembles as being data collections that

- always cover a certain phenomenon,
- combine a set of single, but related, individual ensemble members,
- show slightly different characteristics among the individual ensemble members, which is due to varying variables affecting the phenomenon, and
- consist of a significantly large number of individual ensemble members.

This **definition** certainly covers the results created by simulations. However, since this definition is rather general, other datasets in visualization may also fall into the same category. This is on purpose, because due to the increased availability of high-performance computing resources and the large variety of data analysis algorithms (e.g., for segmentation [KAC+12]), other application areas (e.g., material sciences [SZ11]) nowadays also have to deal with similar visualization problems. For example, in material sciences, it is often necessary to find the best segmentation strategy for Computed Tomography data of a given specimen. In this case the ensemble describes possible segmentations of the specimen (*studied phenomenon*). The ensemble then contains different segmentations (*ensemble members*), which originated from segmentations with different

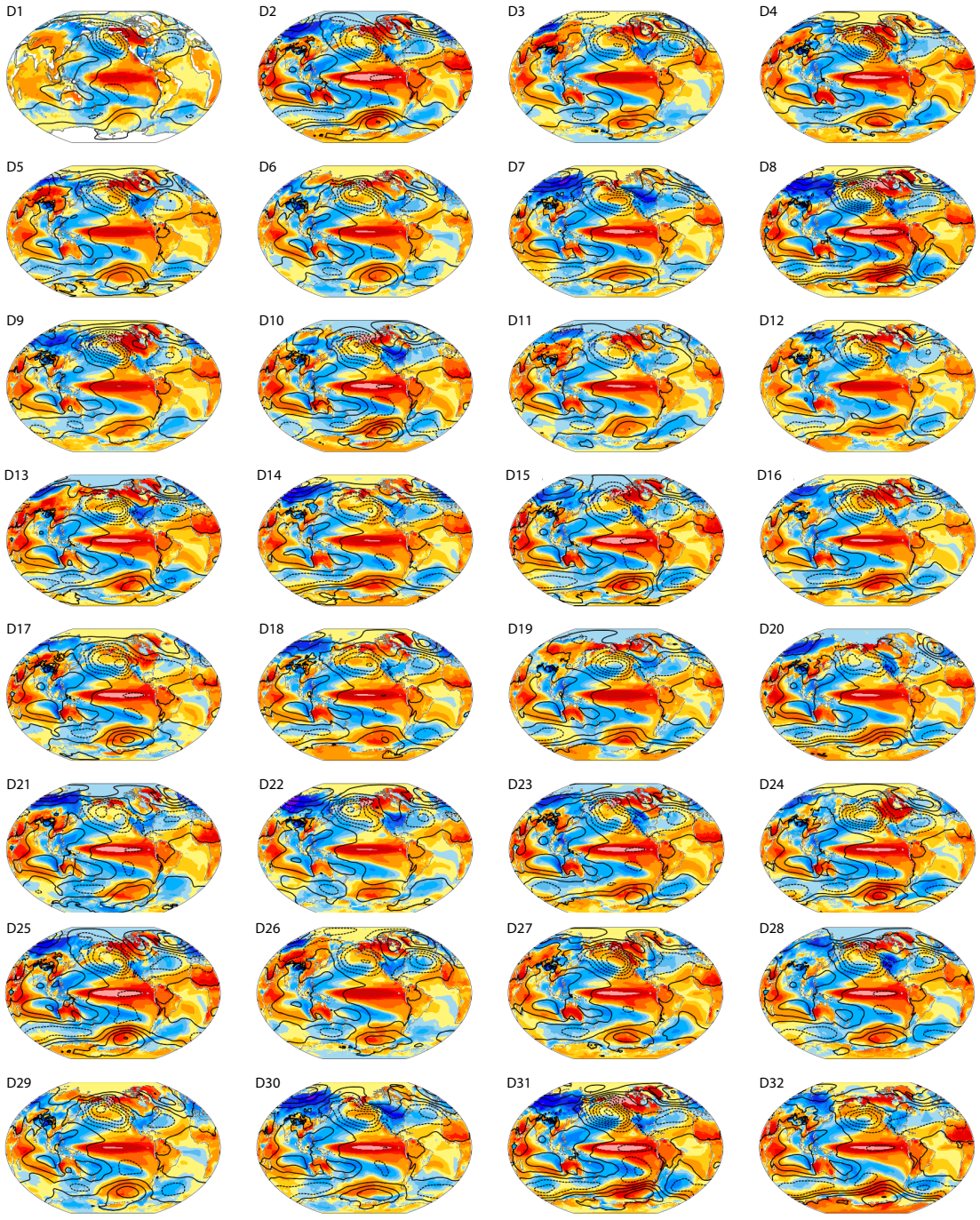

Figure 1.4: Ensemble datasets. This is an example for an ensemble dataset from the weather and climate simulation domain. It shows climate variability in models and observations of the El Niño phenomenon [(NC14].

algorithms or parameter settings (*slightly different characteristics*). Typically, a lot of such segmentations have to be created to find the best solution (*large number of ensemble members*). Since such datasets are becoming more popular nowadays, ensemble visualization is an emerging topic in visualization. This thesis therefore concentrates not only on ensembles originating from the simulation domain, but also on other datasets showing the same characteristics.

Before the definition of the term ensemble, people also used **different expressions** for what can now be called an ensemble dataset. Hansen et al. [HCJ+14] referred to ensembles as *multi-field* data. Love et al. [LPK05] introduced the term of *multi-value data*, describing spatial data that consists of a high-dimensional data vector at each spatial location in the domain. They proposed either a parametric or an operator approach to visualize this kind of data. The parametric approach analyzes the whole dataset, assuming that the data vectors can be adequately described by statistical parameters. The operator approach visualizes the multi-values themselves, by using visual concepts such as streamlines or iso-surfaces. *Multi-variate data* describes data that contains several different connected data types (e.g, scalars, vectors, tensors) that need to be integrated into one visualization [FH09]. The term *multi-modal* data is used if different modalities of the same phenomenon exist and need to be analyzed together (e.g., body tracking, audio, and gaze of one patient [WAC+13]). Similar to this, *multi-model* data describes how results from different computation models (e.g., a climate model and a fluid-physics model) that share certain parameters (e.g., temperature) can be merged in a simulation and in the following analysis [KMDH11]. Another aspect that has lead to its own research branch in visualization is the domain of *time-varying data*. In this case also mostly the same phenomenon is observed over time, and changes are recorded in the visualization [AMST11]. Another domain also dealing with ensembles is the visualization research branch of *parameter space visualization* [SHP+14]. In this case researchers propose methods how the parameter space spanned by the ensemble can be explored and analyzed. Closely related to ensembles in the weather and climate domain, the term of *uncertainty visualization* has been created [HCJ+14]. Uncertainty visualization relates to the fact that due to the multitude of information in an ensemble, decisions on this data can only be made based on statistical probabilities. The success of uncertainty visualization was, again, greatly driven by the analysis tasks needed in the weather and climate simulation domain. In this thesis we will continue to use the term ensemble, and we will also mainly concentrate on the visualization of such datasets.

Due to the large number of ensemble members, and due to the complexity of the data, ensembles rely on a closely coupled human and machine interaction for their analysis [TC05]. Ensembles exhibit great advantages for a automatic or semi-automatic analysis of phenomena. First of all, the ensemble members describe the same phenomenon under different prerequisites. The ensemble members are therefore defined in the same spatial (and temporal) domain and typically are of the same data type (e.g., volumetric or vector field data). Furthermore, the ensemble members are usually co-registered and of the same size, which allows for comparisons amongst them. In **ensemble visualization**, researchers work on new techniques how ensembles can be visually analyzed in an automatic or semi-automatic way. They often make use of concepts from comparative visualization, which then need to be extended to the large amount of data available in ensembles. While comparative visualization techniques can also be applied to

just two objects, ensembles require a different approach for analyzing them. Due to the large amount of data, the raw data values can be hardly presented to the user, since this would result in over-plotting in any standard visualization system. Alternatively, in ensemble visualization it is typically required to use aggregation methods (e.g., statistical measures or clustering) to reduce the dimensionality of the available data, and provide overviews to the users. It is important that the aggregation techniques encompass a set of underlying distributions, because simple techniques (e.g., summation) will suppress small features that might be of interest for the analysis. So far, it was not possible to find an overall visualization that would cover all cases [WP09]. Instead, many people advocate the use of multiple linked views, where each of the views can convey a different facet of the data [LSP+10]. Another advantage of multiple linked views is that representations the experts are already familiar with can be re-used in the analysis, and can be combined with standard plots and novel visualization techniques.

So far, according to Obermaier and Joy [OJ14], the existing techniques for ensemble visualization can be divided into the two groups of **feature-based** and **location-based** visualizations. *Feature-based* visualizations, as the name already implies, extract features from the ensemble members, and then compare these extracted features across the whole ensemble. Such features can, for example, be iso-surfaces in a volume dataset, or clusters in an abstract dataset. Feature-based techniques imply that it is possible to extract such features from all the ensemble members, and that a comparison metric exists that allows for the comparison of the features in the ensemble. Further, appropriate visualization techniques are needed to visualize the differences of the features across the ensemble. *Location-based* visualizations, on the other hand, concentrate on fixed locations in the ensemble and compare ensemble members at these fixed positions. Such positions can either inherently exist in the data (e.g., geographic regions in case of weather data), or can be defined by the user. The more abstract the ensemble data, the more the techniques usually shift towards feature-based visualization, because then spatial locations in the data are not that important anymore, or do not even exist.

In this thesis a special focus lies on the usage of **location-based techniques** for ensemble visualization. Interestingly, a global (mostly statistical) evaluation of the ensemble members alone does not communicate a full understanding of individual dataset features in all cases. It might happen that individual datasets with a low overall statistical significance contain data features in certain local regions, which are not desired by the users. For example, when working with 3D for archaeological preservation purposes, it is important that small details like stucco work are preserved in the data [BHE+15]. Algorithms for mesh reconstruction from point clouds, or algorithms for mesh denoising, sometimes have the tendency to smooth the data. Although the results of such algorithms might not show any global statistical significance, because they handle noise quite well, edges and corners are smoothed in the result. This would not be a desired behavior for archaeological purposes, because then the small details like stucco work would be lost. In this case experts might prefer the result from another, more noise-sensitive, algorithm, which also better reconstructs small features. A comparison of a smoothing and a detail-preserving mesh denoising can be seen in Figure 1.5. To reveal such cases, it is necessary to allow the users to concentrate on certain local regions of interest in the data. A dataset containing different results from one mesh editing algorithm (with different parameter settings), or results

Figure 1.5: Local details analysis. Global statistical evaluation of ensemble members may hide information about small details in the data. In this example two mesh denoising algorithms for reconstructions of a laser scan of a *statue* can be seen. As the *close-ups* reveal, an algorithm *smoothing* the data cannot *preserve small details* in the data [LWZ+16].

from different mesh editing algorithms applied to the same input data, can be generalized into the already defined concept of ensembles. Since such ensembles are typically complex (with regards to data dimensions), sophisticated interaction techniques are needed that let users define regions of interest in the data [EF08]. These definitions of local regions can be done by different interaction concepts. In case the data has a geographical component (e.g., weather and climate data), users might want to select regions of interest simply based on geographical properties (e.g., latitude/longitude, name, etc). For non-geographical, or even abstract data, there are no pre-defined locations available that could be harnessed by the users. In the case of complex 3D data, the semantics behind a user's definition of a region of interest has to be defined more accurately, because users are always operating on the data within a certain context. The context is defined by the current viewing position, and by other parameters influencing the representation (e.g., a transfer function). Interaction techniques, therefore, have to be aware of the context, so that they can correctly interpret a user-based selection [WVFH12]. Another challenge for location-based techniques is not only to let users define regions of interest, but also to allow the users to explore these regions further. Due to the complexity of ensembles, it is a challenging task to provide more information about a local region of interest in a concise way. In many cases multiple coordinated views are employed to show more information about a user-defined region in the data [KH13]. Another possibility is to use in-place techniques [MBS+12] to show more information about the underlying data at the spatial position of the region of interest. The choice of the visualization concept also highly depends on the ensemble data, and on the tasks that have to be solved by the users.

1.2 Related Work

Researchers in *comparative visualization* work on novel techniques how objects can be visually compared. Existing approaches for comparative visualization are reviewed in Section 1.2.1. Due to the availability of ensemble datasets, analysts are faced with the problem of applying comparative visualization techniques to large amounts of data. The related techniques and applications targeted to *ensemble visualization* are reviewed in Section 1.2.2. To reveal local differences in datasets, interactive *local data exploration* techniques are needed, that allow the users to concentrate on regions of interest in the data. Such techniques are discussed in Section 1.2.3.

1.2.1 Comparative Visualization

This section reviews existing approaches for comparative visualization. The presented techniques are related to the taxonomy of the three groups of *juxtaposition*, *superposition*, and *explicit encoding*, as proposed by Gleicher et al. [GAW+11].

Juxtaposition means that objects can be placed side-by-side to facilitate comparison. The comparison of the objects and the detection of differences is done by the users. This concept has already been used by Tufte [Tuf86], which he called the concept of *small multiples* at that time. An example for the usage of small multiples to compare vessel movement data can be seen in Figure 1.6. Later *scatter plot matrices* were used to analyze multidimensional data [BSM+13]. Adding juxtaposition to an existing visualization system is rather easy, because it does not require changes in how objects are drawn. Therefore, juxtaposition is used as a comparison concept in many applications, often referred to as *side-by-side views* [AH11], or *dual views* [NSGS07]. Munzner et al. [MGT+03] presented *TreeJuxtaposer*, a technique that allows biologists to compare large phylogenic trees by placing the information side-by-side. Verhagen and van den Berg [VvdB08] used juxtaposition to compare nutrient profiling schemes. Lampe et al. [LKH10] arranged different abstractions of a large dataset in side-by-side-views, so that users could spot temporal patterns. Hotz et al. [HSNHH10] used juxtaposition to compare diffusion tensor fields. Juxtaposition can also be used to compare graphs, as shown by the work of Burch et al. [BVB+11]. They arranged graphs vertically in a way that the edges are all directed from left to right, so that they can be compared to each other.

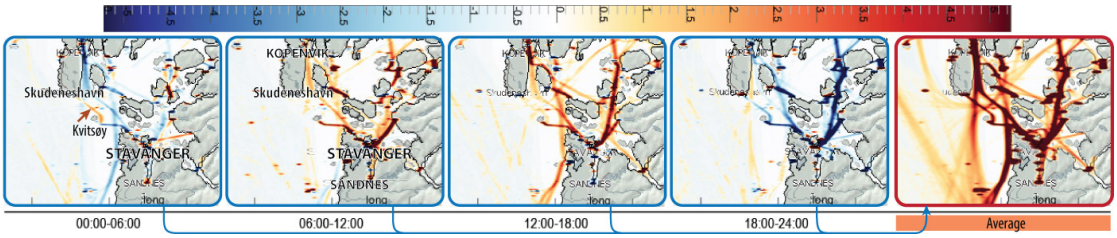

Figure 1.6: Example for a *juxtaposition*. Comparison of vessel movement data for different time spans and weekdays in a small multiples display [KPBG13].

In the case **superposition** is used, objects that need to be compared are placed in the same view, and in the same coordinate space. Depending on the data, different strategies have to be used to present several objects at once. *Blending* (i.e., making objects semi-transparent) is a common method to place images in the same view. Kammerer et al. [KHZ04] used this method to spot differences between infrared and color images of ancient paintings. More advanced techniques for the superposition of images would be *color weaving* [HSKIH06], or *attribute blocks* [Mil07]. Malik et al. [MHG10] proposed an approach for comparing images. Their technique subdivides the image space into hexagonal regions, and each region is subdivided into smaller elements which depict data from different series. This way it is possible to compare different volume datasets in a slice-based way in one view. The technique is shown in Figure 1.7. If graphs should be compared, it is possible to use color coding or strokes to encode the different objects in the same view [EKLN04]. Jianu et al. [JYC+10] employed superpositions for graphs to compare different proteomic pathways. A challenging task is the comparison of 3D data, due to the large amount of data, and due to the possibility of occlusions. Alabi et al. [AWH+12] proposed to support the comparison of 3D surfaces by placing slices of the surfaces in the same view. The slices of the different surfaces are interleaved, so that they can be compared with each other.

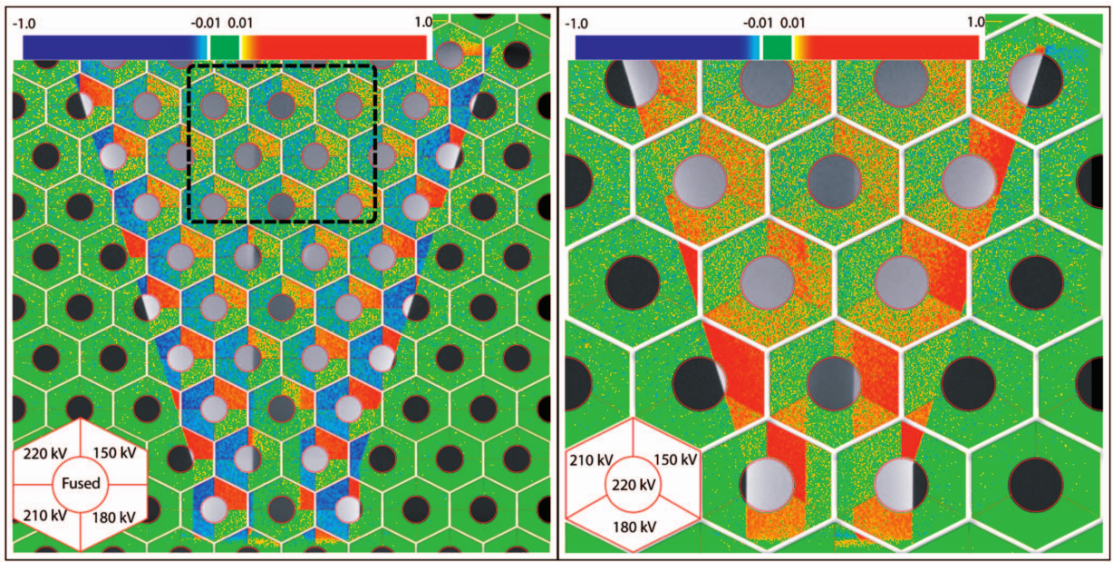

Figure 1.7: Example for a *superposition*. Slices of volume datasets can be compared by subdividing the image space into hexagonal regions. Each of these region is then subdivided into smaller elements that depict data from different volume slices [MHG10].

So far the identification of relationships between different objects and the detection of differences was done by the users. If using **explicit encoding**, the users are pointed to where to find the differences in the data. This implies, though, that the differences can be computed by some metric. Volume datasets, for example, can be compared by computing the differences in a voxel-based way, and by explicitly encoding the differences through using surfaces [WS06]. Another possibility is to encode the differences through using graphs [WSKK06]. Similar to

this, Sauber et al. [STS06] visualize correlations between 3D multi-field scalar datasets. For diffusion tensor volumes, differences can be encoded by color [DZDL02] or by glyphs [ZSL+16]. As an example for non-spatial data, Tory et al. [TSFH+13] showed how changes in construction schedules can be compared. There are some examples of how explicit encoding can aid the comparison of 3D surfaces. Some approaches just use color encoding [MIA+03], while others also employ additional symbols like arrows [WT05].

The three concepts of juxtaposition, superposition, and explicit encoding (or just a subset of them) can also be combined into **hybrid solutions**. In this case the strengths of several concepts can be combined into one design. A very common technique is to combine one of the concepts arranging objects in a spatial way (juxtaposition or superposition) with explicit encoding. In this way objects are put in relation to each other, and the differences are also clearly visible. Drucker et al. [DPA06] used a combination of juxtaposition and explicit encoding with connecting lines to compare different versions of a *PowerPoint* presentation. The same combination of concepts can be used to compare two networks, by showing the color-coded differences in the middle between the two graphs that are compared [AWW09]. The combination of superposition and explicit encoding may be redundant, because the superposition of the objects already reveals the differences among them. In many cases explicit encoding is therefore used only to emphasize differences in the data. Another possibility is to use an aggregated view or a summarization of the objects, and to display the differences on top of it [EST07]. Busking et al. [BBF+11] used superposition to compare surfaces in 3D, and in addition used glyphs to enhance the visualization of the differences (see also Figure 1.8).

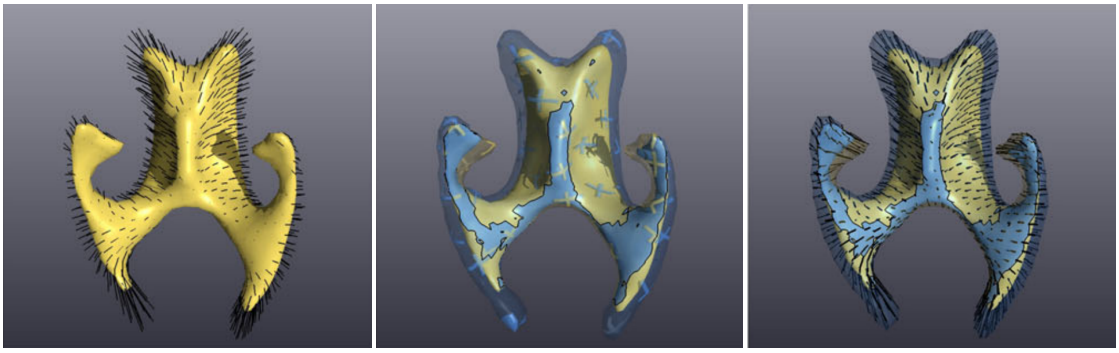

Figure 1.8: Example for a *hybrid solution*. Two surfaces in 3D are compared by placing them in the same 3D view (superposition), and by enhancing the differences among them with glyphs (explicit encoding) [BBF+11].

Apart from the three categories, other visualization techniques can be used to support comparisons. A very important concept for analyzing datasets is **interaction**. The users can employ drill-down techniques to interactively focus on interesting parts in the dataset [VJC09]. Another interesting concept to convey changes is **animation**. Keefe et al. [KERC09] showed how animation can be used to visualize biomechanical motion data, and Hermann et al. [HSSK16] used animation to convey differences between volume datasets.

1.2.2 Ensemble Visualization

In this section existing approaches for ensemble visualization are reviewed. The techniques are divided into the two categories of *feature-based* and *location-based* approaches, as proposed by Obermaier and Joy [OJ14].

Feature-based techniques first extract features from the raw data that can then be compared across all ensemble members. Ensemble visualization was greatly driven by application cases in the weather and climate simulation domain, therefore much research was done in this area. The system *Met.3D*, as introduced by Rautenhaus et al. [RKSW15], can be used to, for example, detect warm conveyor belt situations in the weather data [RGSW15]. These are important informations for airline companies. Höllt et al. [HMZ+14] presented *Ovis*, a system to analyze the different sea surface heights for ocean forecasting. Diehl et al. [DPD+15] provided means to search for patterns in weather ensembles. They made use of small multiples and curve-comparison techniques to help users find patterns in the data. Also other domains already employed ensemble visualization techniques. Beham et al. [BHGK14] developed a technique combining parallel coordinates and glyph-based visualization to analyze a set of simulation results from a cup generator, an algorithm that generates 3D models of cups. The visualization helps to find proper parameters settings for the generator. Piringer et al. [PPBT12] proposed a framework for the visual analysis of simulation data from the automotive industry. A very important technique for feature-based ensemble visualization is the extraction of contours or iso-surfaces from the data. A common way to present the features in 2D is to use *spaghetti plots* [PWB+09]. In a spaghetti plot all available contours/lines are drawn on top of each other, but in different colors. Spaghetti plots are a useful tool to get an overview of the available data, to identify patterns or clusters, and to detect outliers in the distribution. However, for larger datasets, spaghetti plots heavily suffer from over-plotting. Therefore, researchers tried to find new techniques how these plots could be improved. Sanyal et al. [SZD+10] invented *Noodles*, where they used ribbons and glyph-based techniques to better convey the content of a spaghetti plot. Mirzargar et al. [MWK14] developed *curve boxplots*, where they show a statistical summary of the spaghetti plot data. The summary included the median curve, an indication where 50% of the curves will result, and outlier curves. Similar to this, Whitaker et al. [WMK13] proposed to arrange contours in a *contour boxplot*. The display of various information is already complicated in 2D when using spaghetti plots, but becomes even more complex in 3D. Due to occlusions, it is basically useless to plot the information from more then one to three ensemble members at once. Ferstl et al. [FBW16] therefore introduced variability plots for streamlines, where users can see the main directions and outliers in a vector field. For 3D surfaces, color-coding the variability [PRW11], or presenting the variability of surfaces in one plot view [MGKH09] can help users to get an overview of a set of 3D surfaces. Genton et al. [GJP+14] proposed *surface boxplots*, where a statistical summary of the available surfaces is presented. This technique can be extended by integrating surface boxplots directly into the visualization [RMK+15]. The median surface, the locations of the other contours and outliers are clearly depicted. An example of such surface boxplots can be seen in Figure 1.9. The display of multiple surfaces in 3D is still not easy to achieve, and is therefore still an ongoing research topic for the visualization of ensembles. Demir et al. [DDW14] presented *MultiCharts*, a system to visualize temperature curves in weather simulation data. They analyzed multiple

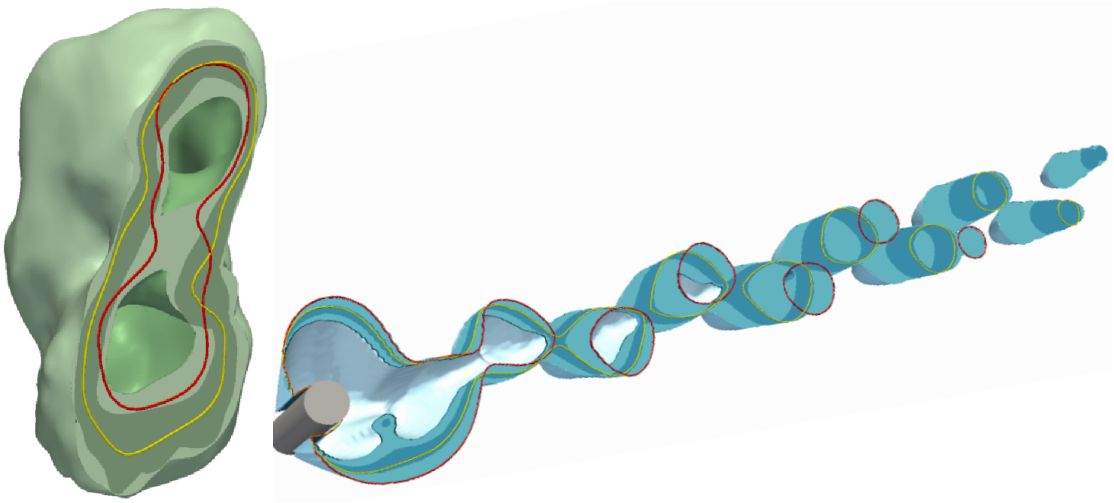

Figure 1.9: Example for a feature-based comparison. The surface boxplot allows users to compare an ensemble of 3D surfaces by showing the median surface contour (yellow) and outlier contours (red) [RMK+15].

volume datasets created by weather simulations. From this data they extracted 3D data points, linearly arranged them along a space-filling curve, and drew them as multiple charts in the same plot area. MultiCharts can be used to quickly provide an overview of the data and depict data regions exhibiting unusual behavior. If a different linearization mechanism is used, the system can be shifted from a feature-based to a location-based analysis tool.

In the case of **location-based techniques**, the visualization supports experts in analyzing ensemble properties at selected spatial positions. For climate data, Böttinger et al. [BPR+15] proposed a technique to visualize different parameters by using color-coding. In their system it is also possible to concentrate on local regions of the data, to detect local anomalies in, for example, the temperature. Waser et al. [WFR+10] concentrated on the visualization of heterogeneous simulation runs for environmental disasters, in particular flooding. Their technique *World Lines* allows users to browse the outcome of the different simulations, but it allows users also to concentrate on local regions. For example, it can be virtually tested how the placement/elimination of a barrage would affect a certain flooding situation. For data from computational fluid dynamics, Hummel et al. [HOGJ13] developed a framework to visually convey how ensemble members agree or disagree in certain regions. If ground-truth data is available, ensemble members can be compared with it and outliers can be quickly identified. This outlier identification can also be done based on local features [GBP+13]. Coffey et al. [CLEK13] proposed *Design by Dragging* for in-place queries on large ensemble data. Their system allows users to locally edit the ensemble data, and adjust the available parameters, so that the final outcome suits the users' needs. A very important concept of the system are the interaction possibilities, so the authors implemented multi-cursor and multi-touch capabilities to locally manipulate the data. Jarema et al. [JDKW15] used a glyph-based visualization to show the local variability of a 2D vector field ensemble. The vector field was created from weather simulations. The users can select regions in the main view,

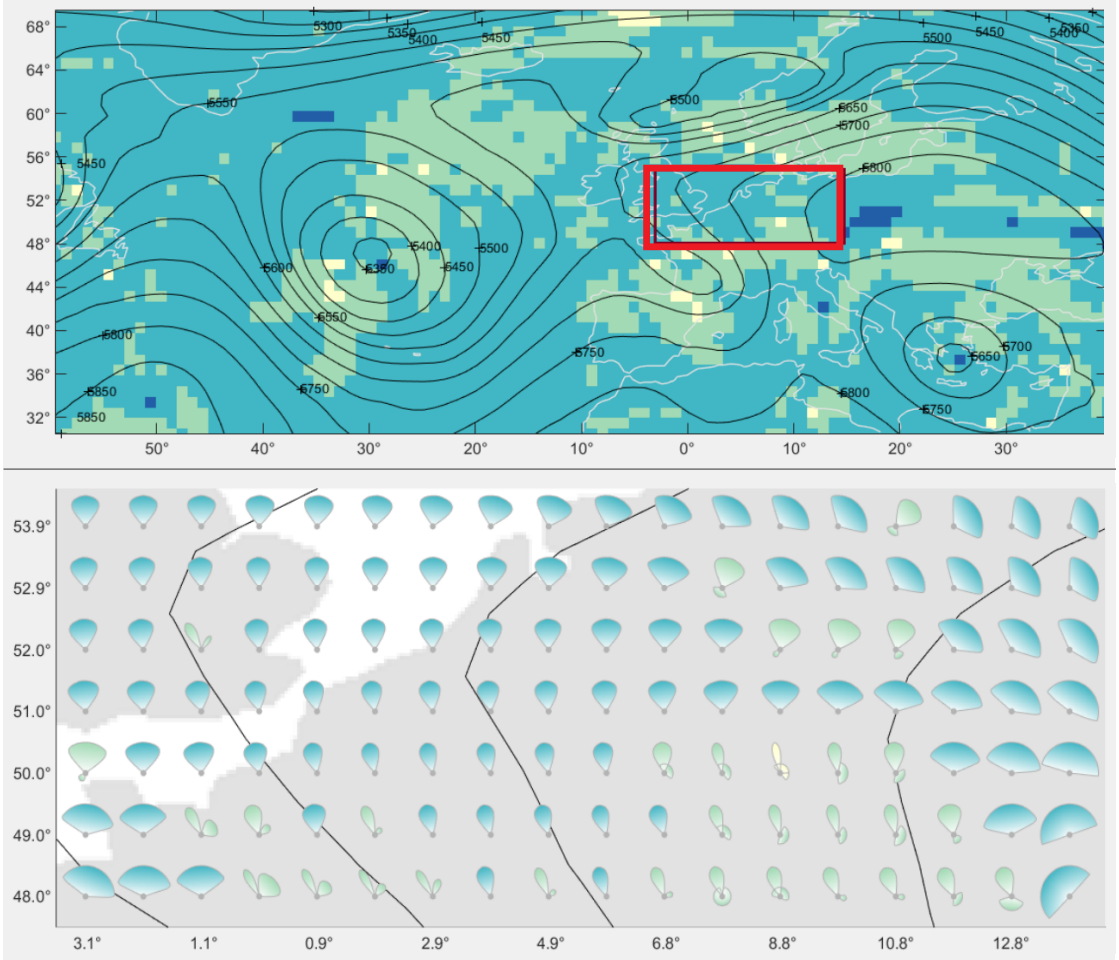

Figure 1.10: Example for location-based comparison. Users can select a region in a 2D vector field ensemble (*upper part, red box*), and then the region is shown in more detail (*lower part*). Glyphs are placed at certain fixed positions in a 2D vector field ensemble to visualize the parameters at these positions [JDKW15].

and then the local parameters of these regions are shown in a multiple-linked-view environment. In Figure 1.10 a user-defined region of interest in the 2D vector field (red rectangle) and the corresponding detail view are shown. In the detail view glyphs are placed on a regular grid to depict the values of the ensemble parameter at the corresponding positions. It is also possible to select positions in the grid, and then the system searches for other locations similar to the selected one. The domain experts evaluating the system highly appreciated the possibility to be able to select spatial regions in the data.

1.2.3 Local Data Exploration

The question of how local regions can be inspected while still having an overview of the overall context is addressed with *Focus+Context* in the literature. Several approaches in 2D and 3D can already be found [CKB09]. In the case of ensembles, a local exploration of the data is especially challenging, since the data is typically very large (in terms of number of ensemble members) and complex (in terms of data dimensions).

If dealing with 2D data likes images or vector fields, it is rather easy to let users define a region of interest by mouse click. In the case of surface data, which is usually defined in 3D, it is also possible to use mouse interaction to define a local region of interest. In the case of complex 3D volume data, or 3D vector field data, the selection of regions of interest is not that straightforward any more, because of the additional aspect of depth. The users in all cases see a 2D representation of the data on the screen. Whenever users employ mouse interaction, it has to be defined which depth inside the data they are actually referring to. In the case of volumes, the process of selecting structures inside the data is usually referred to as **volume picking**. It is possible to decide which depth to take by analyzing the density profile along the ray that is defined by the mouse click and the current viewing direction [KBKG09]. Selected regions of interest can be used in a visualization system to display additional information about the volume [RVB+09] in interactive close-up views. The technique was implemented for the purpose of medical reporting, and each closeup could be used to show different modalities using different visualization styles. Mlejnek et al. [MEV+05] implemented *Profile Flags*, a volume probing technique to reveal structures in volumetric datasets and display them along an additional axis in the 3D environment. Volume probing is not only useful to display more details about a certain region in a volume, but can be also used to improve the current depiction of the volume. Transfer function design [KKH02] can be greatly enhanced by letting users select regions of interest in a volume dataset. The selection of visible structures can be used to refine the segmentation of certain parts in the volume [SK07]. Picking structures in a volume can also be used for a successive interaction. Patel et al. [PBVG10] implemented a system for analyzing volumetric seismic data, where users can interactively assemble horizon parts by picking.

When analyzing **flow or vector data**, experts are often interested in the local parameters of a flow field, or how local regions in the data contribute to the overall situation. De Leeuw and van Wijk [dLvW93] created probes that can be placed in a flow field to study the local parameters. Flow parameters like the velocity and the velocity gradient tensor (i.e., the local change of velocity) are then mapped to geometric primitives in the visualization. Isenberg et al. [IEGC08] proposed a similar technique for 2D vector fields. Glyphs are placed in the view which then represent the parameters in specific local regions. Wiebel et al. [WGS07] suggested an approach to find out how much certain subsets of the data contribute to the overall flow. They used probes which can be placed by the user to define these subsets, and then the visualization reveals the subset has on the rest of the flow dataset.

1.3 Contributions of this Thesis

The contributions of this thesis are based on research papers, which are described in more detail in the following (Chapter 2, Chapter 3 and Chapter 4). A summary of the contributions is given in Section 5.1.

Contribution 1 - VAICo

VAICo: Visual Analysis for Image Comparison [SGB13] presents a visual analytics pipeline for the interactive analysis of image ensembles. We compute the pixel-wise differences in the data and present them to the user. Then we provide drill-down interaction possibilities, so that users can find out what the differences are, and which members of the ensemble caused them. We applied our technique to different image datasets from different domains, and it turned out that VAICo clearly helps to identify patterns in the data.

Contribution 2 - YMCA

YMCA - Your Mesh Comparison Application [SPA+14] describes an application suited for the comparison of an emsemble of 3D meshes. We analyze the data and then visually encode regions of high variance in the data. Such high-variance regions are of special interest, because they point to spatial locations where the meshes exhibit different results. The users can employ a magic-lens widget to explore local regions in the data. Interesting regions exhibiting a high variance are arranged in a parallel coordinates plot, where the different ensemble members can be compared based on local spatial regions. It is also possible to add new user-defined regions to the plot. The YMCA system was applied to data from point-cloud reconstruction, and we identified patterns in the data that would be tedious to detect otherwise.

Contribution 3 - VALENE

Visual Analysis of Volume Ensembles Based on Local Features [SFP+16] aims at the analysis of local regions in a volume ensemble dataset. As an exemplary use case, we concentrated on volumetric segmentation masks for this work. The users can specify local regions of interest by placing volume probing widgets in 3D. The depth of the widget placement can be selected by positioning a slicing plane in 3D. The placed widgets are arranged in a graph based on the similarity of the local ensemble data. Using the graph information, similar local regions can be detected inside the ensemble. It is further possible to compare individual ensemble members against the rest of the ensemble.

CHAPTER 2

Visual Analysis of Differences in Image Ensembles

This chapter is based on the following publication:

Johanna Schmidt, M. Eduard Gröller and Stefan Bruckner. VAICo: Visual Analysis for Image Comparison. *IEEE Transactions on Visualization and Computer Graphics*, 19(12), pp. 2090–2099, IEEE, New York, NY, USA, December 2013

2D images are used in a large variety of domains to view, analyze and present results of different tasks. The comparison of images is therefore an important task in data analysis. In the medical domain, for example, deviations in slices of magnetic resonance or sonographic imaging datasets can indicate anomalies which should be further inspected by the domain experts. In image processing, results of different edge detection or segmentation algorithms have to be compared. In visualization and rendering, resulting 2D images have to be compared with each other to evaluate variances that are caused by different parameter settings. These examples show that there is definitely a demand for comparative visualization techniques for 2D images. The increased availability of analysis algorithms and computing resources also lead to the creation of very large image datasets. Therefore, another important issue for comparing images is scalability pertaining to dataset size (i.e., number of items in the dataset). In biology, for example, datasets are often based on the analysis of several hundreds of specimens. If the images in the dataset describe the same phenomenon, we can then talk about them as image *ensembles*. A very common approach for image comparison is to place them side-by-side or in multiple views, or to overlay them in a semi-transparent way (*blending*). However, due to limitations in the human perceptual capacity as well as due to limited screen space, such comparative visualizations do not scale well. These tools are only suitable for comparing a limited number of images. Another important issue when developing comparative visualization techniques is how to provide information about the underlying original data. In many approaches differences between datasets are mapped to visual

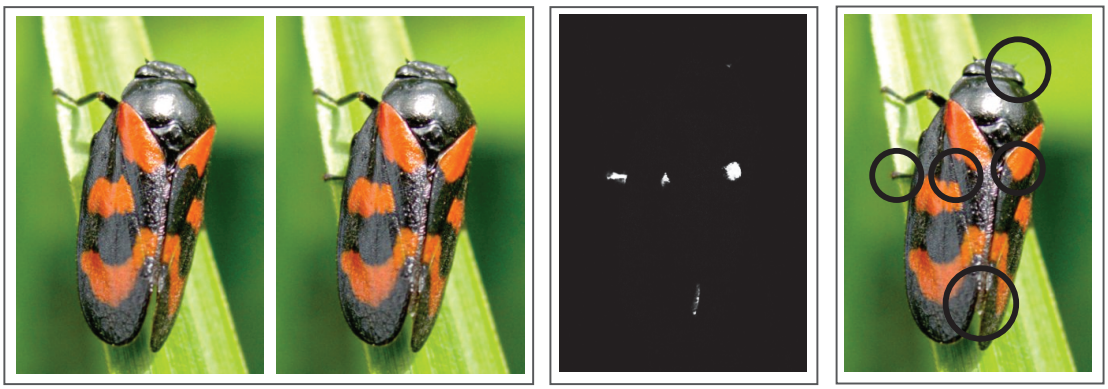

Figure 2.1: Image comparison. This figure shows two pictures that look similar, but in fact exhibit local changes. A very common way to compare them is by placing them *side-by-side*. To help users to find the variations more quickly, a *difference image* can be computed. In this illustration information about the similar parts of the data is lost. Another possibility is to *highlight* differences by certain patterns (e.g., colored circles). In this case similar parts of the data are still visible; however, no further information is provided on how the differences are structured.

attributes such as colored patterns (Figure 2.1). Although this clearly highlights differences and similarities between the datasets, it hides the original data that has been used for the calculation. Having knowledge about the original data allows us to identify patterns in the datasets (e.g., to detect outliers).

A strong focus of our work for image comparison lies on the interactive visualization tools, that provide insight into the underlying raw data. We believe that this can lead to a better overall understanding of the studied datasets. The comparative visualization approach which is introduced in this chapter preserves contextual information, while allowing for a detailed analysis of the variations between datasets. Our approach provides effective means for examining local differences in a large image dataset. It supports users in gaining a better overview of different image characteristics and allows them to further investigate individual local regions.

The main features of our approach are:

- *Scalability*: Unlike previous approaches, the proposed visualization technique is specifically designed to compare image ensembles. It is scalable to a large amount of images.
- *Focus+Context*: Our comparative visualization approach provides an overview of the image differences (i.e., how much of the image space is affected) and allows users to drill-down on individual features. With the drill-down techniques, users can explore the underlying raw image data.
- *Flexibility*: Our approach is not targeted to a certain type of image and is not tied to any particular image comparison metric. We demonstrate this by applying the approach to different image datasets.

2.1 Related Work

The VAICo system is strongly related to comparative visualization (as introduced in Section 1.2.1) and ensemble visualization (as described in Section 1.2.2). There also other applications specifically dealing with the **comparison of 2D images**. For analyzing different light intensities in renderings, Pang and Freeman [PF96] used color and other parameters like textures to highlight differences. Baudrier and Riffaud [EBR07] introduced an approach for comparing ancient documents, mainly by placing them side-by-side and highlighting the differences. Eler et al. [ENP+08] proposed a method to visually analyze image collections. Their visualization method allows feature-based grouping and classification of images, but does not provide means to further inspect individual features. Many approaches use color to indicate differences between 2D images (i.e., *explicit encoding*). Since this is a very simple and intuitive way to display differences, it can be applied to various domains. Hollingsworth et al. [HRTV06] used a specific colorization scheme for difference images to compare 2D gas chromatographies. Sahasrabudhe et al. [SWMJ99] used color coding to highlight differences between visualization results. Apart from color-coding, other methods of abstraction have been used to analyze image differences. As described in Section 1.2.1, Malik et al. [MHG10] proposed superposition for the comparison of images. The images in this case were slices of volume datasets. The image space is subdivided into hexagonal regions, and each region shows data from different volume datasets. This way contextual information is provided about the data, and outliers can be spotted very easily. The more elements a dataset consists of, the more sub-elements have to be created for every hexagonal region. At some point the sub-elements will be too small for a proper analysis, which makes this method unsuitable for large image ensembles. The authors also state that their approach is targeted to grey-scale values only. Our approach for comparative image visualization is somewhat similar to the multi-image view by Malik et al., since our approach also aims at preserving information about the underlying data. Due to the use of clustering, our method is scalable to large image ensembles. Our visualization technique is also only applied to regions where changes take place, and therefore provides a better localization of the differences in the data.

Hierarchical clustering is a statistical method of cluster analysis which aims at building a hierarchy of clusters [DH73]. Hierarchical clustering either follows an *agglomerative* (bottom-up) or a *divisive* (top-down) approach. In the case of agglomerative hierarchical clustering, each object starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. Divisive hierarchical clustering starts with one big cluster containing all objects, and splits are performed recursively as one moves down the hierarchy. Hierarchical clustering has become a de facto standard for analyzing biological gene expression data in the past years [ESBB98]. It is also used in other domains, for example, to analyze audio data as described by Clarkson and Pentland [CP99], or to classify ocean colors as proposed by Yacoub et al. [YBT01]. We use agglomerative hierarchical clustering to embed differences in the set of images in a hierarchy to identify different levels of data variances.

2.2 Visual Analysis for Image Comparison

To effectively convey information about differences in an image ensemble, we decided to provide an interactive visualization system that preserves contextual information while enabling the inspection of individual *image differences*. Our **V**isual **A**nalysis for **I**mage **Co**mparison (VAICo) uses a mixed approach of *superposition* and *explicit encoding*. Our work is focused on the comparison of image ensembles (i.e., up to hundreds of ensemble members). None of the images has to be defined as a reference image. Image differences are interpreted as variations in the image dataset (i.e., local color changes). In particular, VAICo is designed to assist users in identifying distinct classes of variations which characterize the underlying phenomena.

VAICo is based on an image comparison step where the image space, defined by the images in the given dataset, is divided into the two parts of *contextual information* and *regions of differences* (RoDs). Contextual information refers to parts of the image space that are the same in all images. RoDs correspond to variations in the image ensemble. The calculation of the RoDs is explained in Section 2.2.1. Image variations are interpreted as color changes between at least two images in the dataset. Image data related to a certain RoD is then embedded in a hierarchy, which enables the classification of changes in the data. The creation of the hierarchy is explained in Section 2.2.2. The results of the image comparison and clustering can be explored with interactive visual analysis tools, which are described in Section 2.2.3. The whole pipeline of the VAICo system is outlined in Figure 2.2.

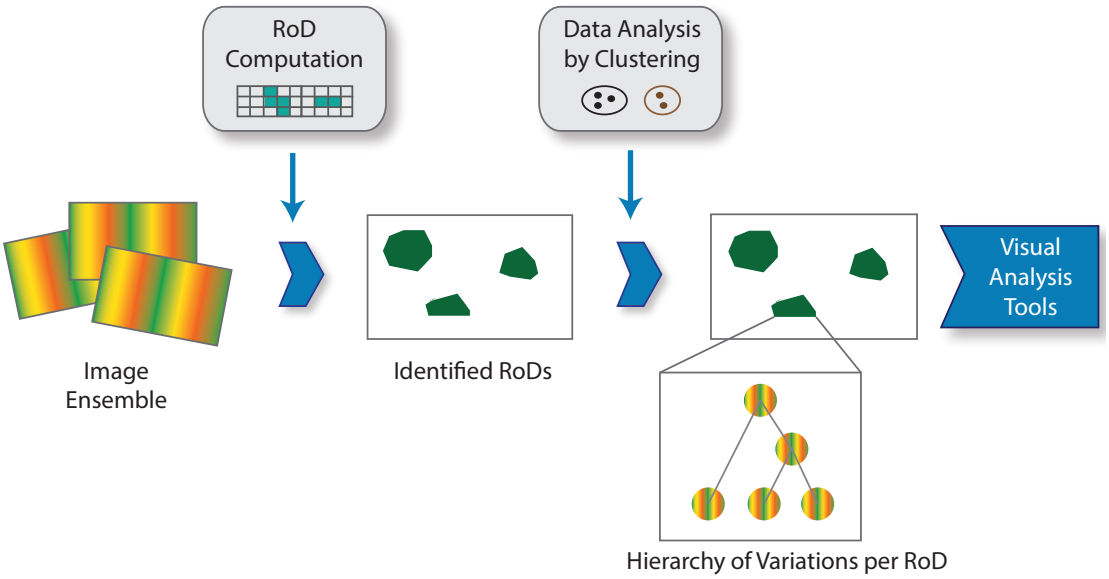

Figure 2.2: Overview of our comparative image visualization approach. VAICo operates in *image ensembles*. To locate regions of differences (*RoDs*), an image comparison metric is applied. In the next step *hierarchical clustering* is performed on the corresponding image data of every RoD. This classifies the differences according to their significance. The results are then presented by using our proposed *visual analysis tools*.

2.2.1 Region-of-Difference Computation

In the first step region of differences (RoDs) have to be identified in image space. We propose two image comparison approaches which both result in a list of RoDs for the given image dataset. We call them the *unbiased*, and the *biased* image comparison.

To apply an **unbiased image comparison**, we employ a pixel-based image comparison metric based on the *Mean Squared Error* (MSE) [ZCW02]. All pixels from one image are compared to pixels at the same location in all other images in the dataset. A threshold is used to filter out low color variations. This enables users to control the algorithm's sensitivity with respect to changes in the data. After applying the MSE calculations to all pixels in image space, a set of *difference pixels* is identified which represent pixels with varying color values in the images. Region growing [SHB98] is then used to group difference pixels together to form disjoint subsets of pixels. Difference pixels are grouped together based on their spatial arrangement. In the region growing approach all difference pixels are considered to be seeds which potentially could start a new region. The region growing is initiated with a randomly selected pixel. Then neighboring pixels (according to an 8-connected neighborhood) are merged into a connected region. The process is iterated until all pixels have been assigned to a region, as demonstrated in Figure 2.3. The subsets resulting from the region growing step define the RoDs which form the basis of our

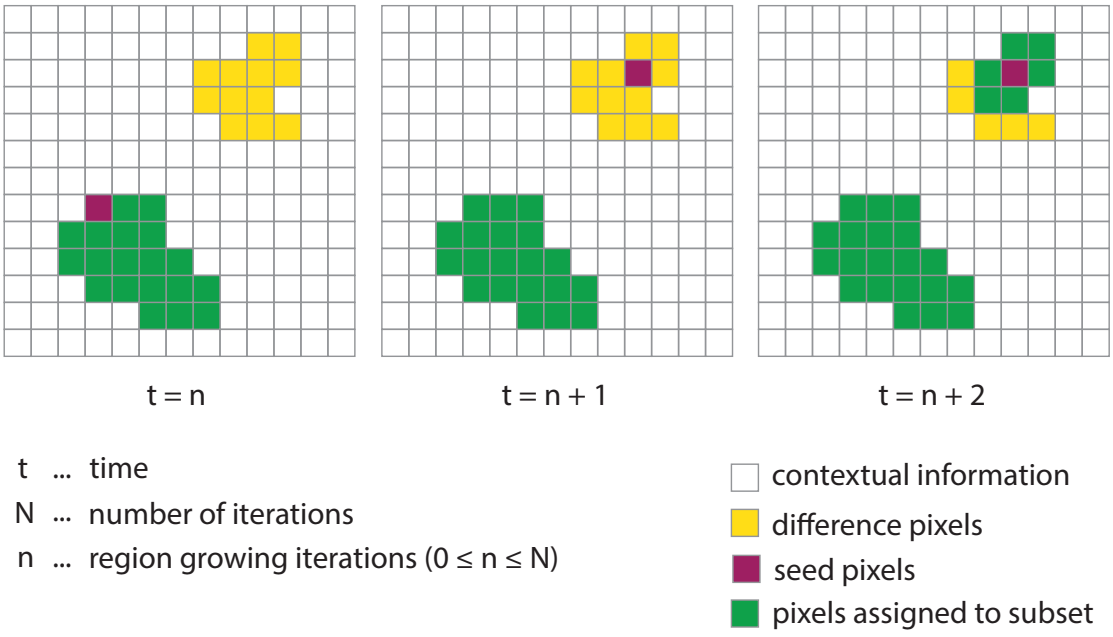

Figure 2.3: Illustration of subset computation by region growing. After the image comparison step a set of so-called *difference pixels* has been identified. Region growing is used to group them together into disjoint subsets. During region growing, difference pixels are assigned to the current subset until it cannot grow any more ($t = n$). Then another difference pixel is selected ($t = n + 1$) and the iteration is continued ($t = n + 2$).

interactive visual analysis approach. RoDs indicate the locations where changes take place in the data. All together, they give an overview on how much of the image space is affected by differences in the dataset and where the differences are located.

In addition to the unbiased image comparison we also propose a **biased image comparison**. This enhances the image comparison by allowing us to include prior knowledge about the data. For some datasets not only image variations are of interest for the analysis, but also differences of *features* in the image. This follows the idea of a feature-based approach for ensemble visualization. Features in this case are represented as connected image regions which can have different characteristics in the image data (i.e., different color values). These features are calculated for every image in the dataset *individually*, before starting the comparison. We employ a color segmentation approach based on *Mean Shift* [CM02] to identify regions of interest in every image in the dataset. The segmented regions then define a set of m features per image in the dataset. We can also refer to these features as RoDs, since they exhibit the same properties (i.e., set of connected pixels). Afterwards image comparison is applied. With the new definition, the image comparison then refers to sets of RoDs being compared to each other. The goal of the comparison is to find out whether RoDs of different images represent the same information. Figure 2.4 illustrates the process of RoD comparison. RoDs are compared based on their spatial position, shape and size. RoDs of different images are considered to represent the same region if their overlap exceeds a threshold (we use 90% of overlapping RoD pixels as a default). The list of RoDs resulting from the biased image comparison describes features that are present in at least one of the ensemble images. Using the biased instead of the unbiased image comparison allows users to eliminate certain regions (e.g., background information) from further analysis.

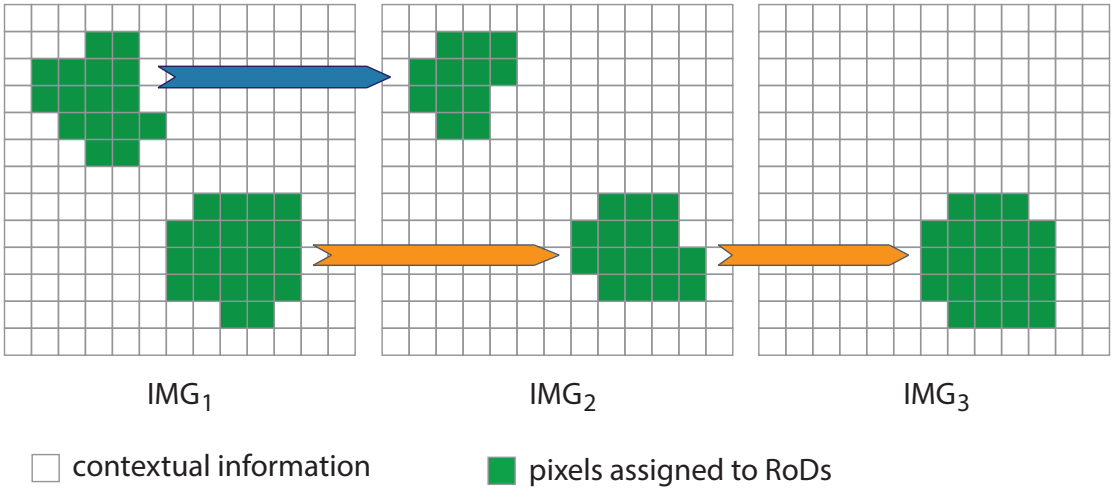

Figure 2.4: Illustration of RoD comparison. After color segmentation a set of RoDs has been defined per image. In the comparison step RoDs of different images are identified that refer to the same information (i.e., overlap). In this figure the upper RoD can only be identified in the first two images (IMG_1 and IMG_2). The second RoD can be identified in all three images (IMG_1, IMG_2, and IMG_3).

Both modalities, the unbiased as well as the biased image comparison, result in a **list of final RoDs**. These RoDs are further analyzed in the next steps to gain more information about the underlying data at the RoD positions.

Since all ensemble images are registered and of the same size, a part covered by a RoD's location and size can be found in every image. A RoD's location and size is defined by its corresponding set of difference pixels. This leads to an unordered list of i image parts per RoD for i images in the dataset. In Figure 2.5 the process for collecting the image parts for a RoD is illustrated. In the case of biased image comparison, only images that contain a certain RoD are further considered. This therefore leads to a list of l image parts per RoD in the case of biased image comparison, where $l \leq i$. The image parts which are collected per RoD from the images are called the **RoD-related image parts** in the following.

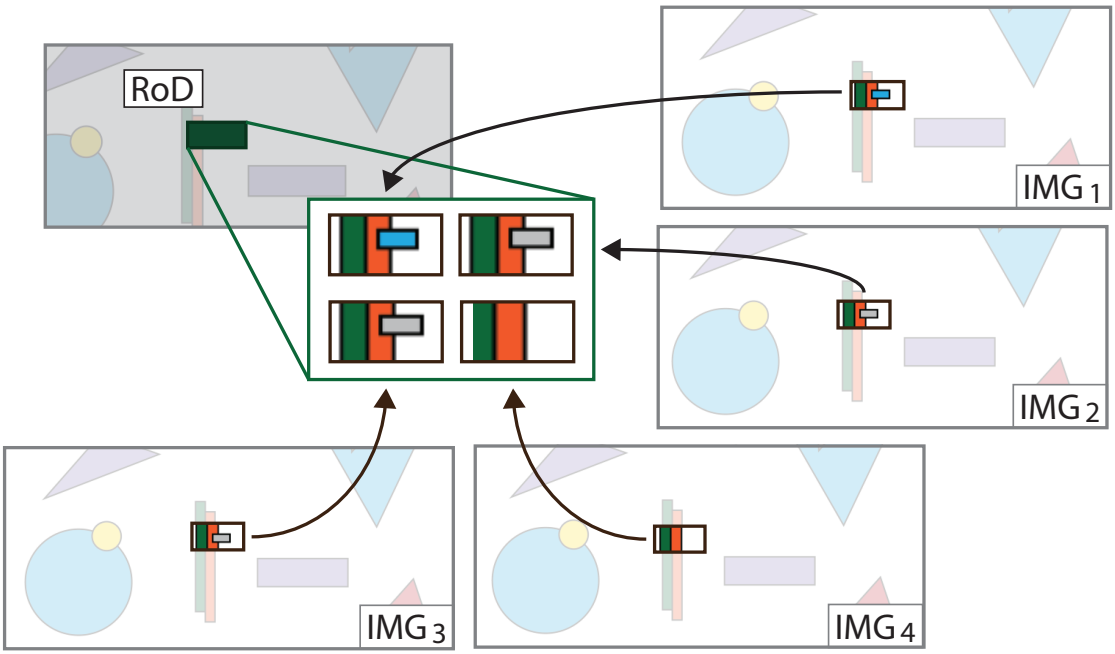

Figure 2.5: Image data corresponding to a certain RoD. In this figure one particular RoD together with its corresponding image parts is illustrated. The image parts are taken from all images in the set (as indicated by the arrows) and are defined by the RoD's size and location (i.e., the corresponding set of difference pixels). The figure describes the collection of RoD-related image parts in the case of unbiased image comparison.

2.2.2 Data Analysis by Clustering

After defining the RoDs and collecting the RoD-related image parts, we now apply clustering to the RoD-related image parts to embed them in a hierarchy. This enables the classification of changes in the data and conveys information about the underlying data.

For the clustering, complete-linkage agglomerative **hierarchical clustering** [DH73] is used. Initially, every RoD-related image part forms a separate cluster. Then in each iterative step clusters are merged together according to their distance. The difference between two image parts is defined by their amount of pixel-wise differences (ranging from 0 to 100 percent). We employ a similarity metric based on the *Mean Squared Error* (MSE). If another image comparison metric should be used to compare the images, this metric could be employed here as well. The clustering terminates once all elements are included in one big cluster. Based on the clustering, a tree is built which describes the results of the hierarchical clustering process (Figure 2.6).

Every hierarchy level in the hierarchical clustering tree describes a valid clustering result for the RoD-related image parts. However, for the best description of the given data, an **optimal clustering** has to be found. In the case of hierarchical clustering, a clustering is basically defined by the number of available clusters. A clustering with maximum accuracy assigns each RoD-related image part to its own cluster. By comparison, a clustering with maximum compression uses one single cluster to include all RoD-related image parts. A clustering is considered to be optimal if it

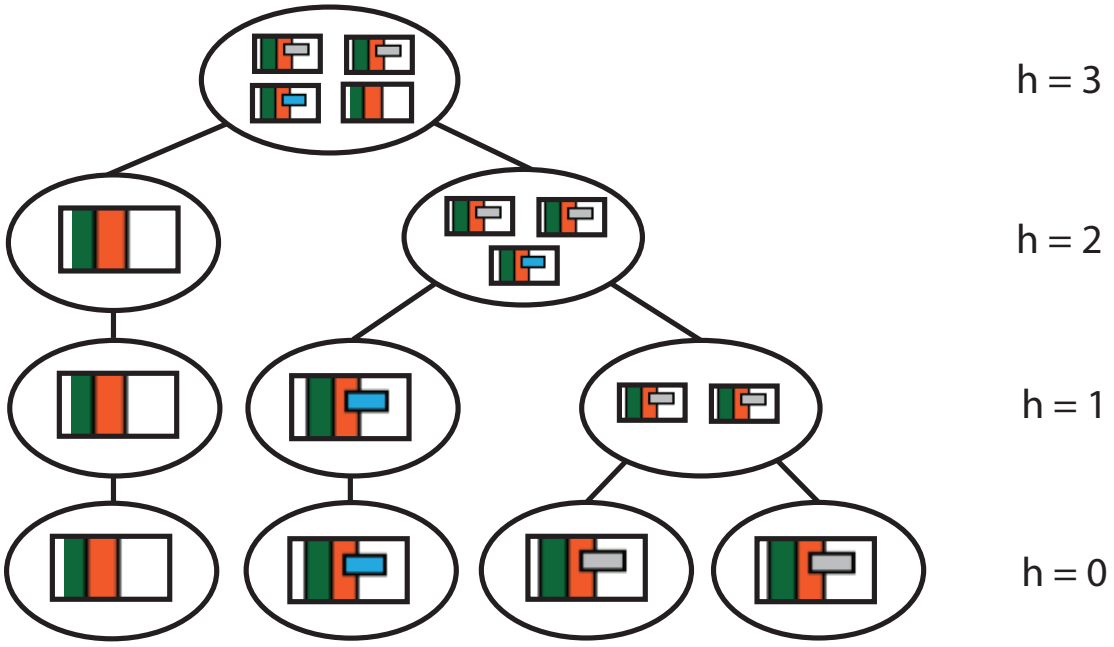

Figure 2.6: Hierarchical clustering result for the RoD introduced in Figure 2.5. h depicts the hierarchy level in the tree. At the beginning ($h = 0$) all image parts are in separate clusters. In further steps ($h = 1, h = 2$) clusters are subsequently merged together according to their distance, until all elements are enclosed in one cluster ($h = 3$).

strikes a balance between the maximum accuracy and the maximum compression clustering. An optimal clustering of the data will classify the image differences and provide more information about outliers as well as similarities in the data. To get an optimal clustering, the hierarchical clustering tree has to be cut at a certain level. We decided to use the *elbow criterion* to determine the tree level which contains the best clustering result. In essence, the elbow criterion specifies that a clustering should be chosen in a way that adding another cluster does not give a better modeling of the data [PK06].

In a clustering, outliers are defined by their distance to other clusters in the set. For our approach **cluster outliers** are of special interest, since they are considered to represent significant changes in the data. Therefore, clusters from the level with the best clustering result are sorted according to their inter-cluster distance. The cluster sorting process is done in an iterative way. In every step, the cluster with the maximum inter-cluster distance (defined by the sum of all distances to other clusters) is determined. It is then stored in an ordered list and excluded from further sorting operations. The process is repeated until no unsorted cluster is left. At the end, an ordered list of clusters is created for every RoD.

2.2.3 Visual Analysis

The results of the image comparison step can be explored with VAICo's interactive visualization tools. Although the main interactions are based on the RoD visualization, it is possible to view the clustering and data analysis results as well. The visual analysis tools of VAICo are illustrated in Figure 2.7.

The interactive visualization elements are embedded in *image space* which is defined by the images in the given dataset. The entire set of images is visualized in one view. Parts of the image space which represent the same information in every image are depicted as **contextual information** (Figure 2.7a). We create an average image of all images in the dataset, and this image is displayed in the background. Pixels are faded out to enhance the visibility of the image variations. This contextual information is needed to embed the interactive visualization tools in the appropriate context.

RoDs are emphasized through **colored polygons** in the foreground (Figure 2.7a). The shape and position of the polygons correspond to the set of difference pixels assigned to the RoD. The RoD visualization provides a visual overview of the image comparison results. In combination with the contextual information, the RoD visualization allows users to immediately differentiate between image variations and regions of similar information. In contrast to marking differences by abstract shapes (e.g., circles or rectangles), the polygon shape provides more information about the extent of the image variations. The number of RoD polygons in image space depicts how many image variations have been identified. The position and distribution of the RoD polygons shows where variations are located in the images, and how much of the image space is affected. In addition to size, shape, and location of the image variations, information about the RoD-related clustering results is also included in the visualization. In case the clustering leads to a higher number of clusters than in other RoDs, the underlying RoD-related image data shows a greater variety than in other cases. This may indicate the existence of outliers in the data, wherefore these cases should

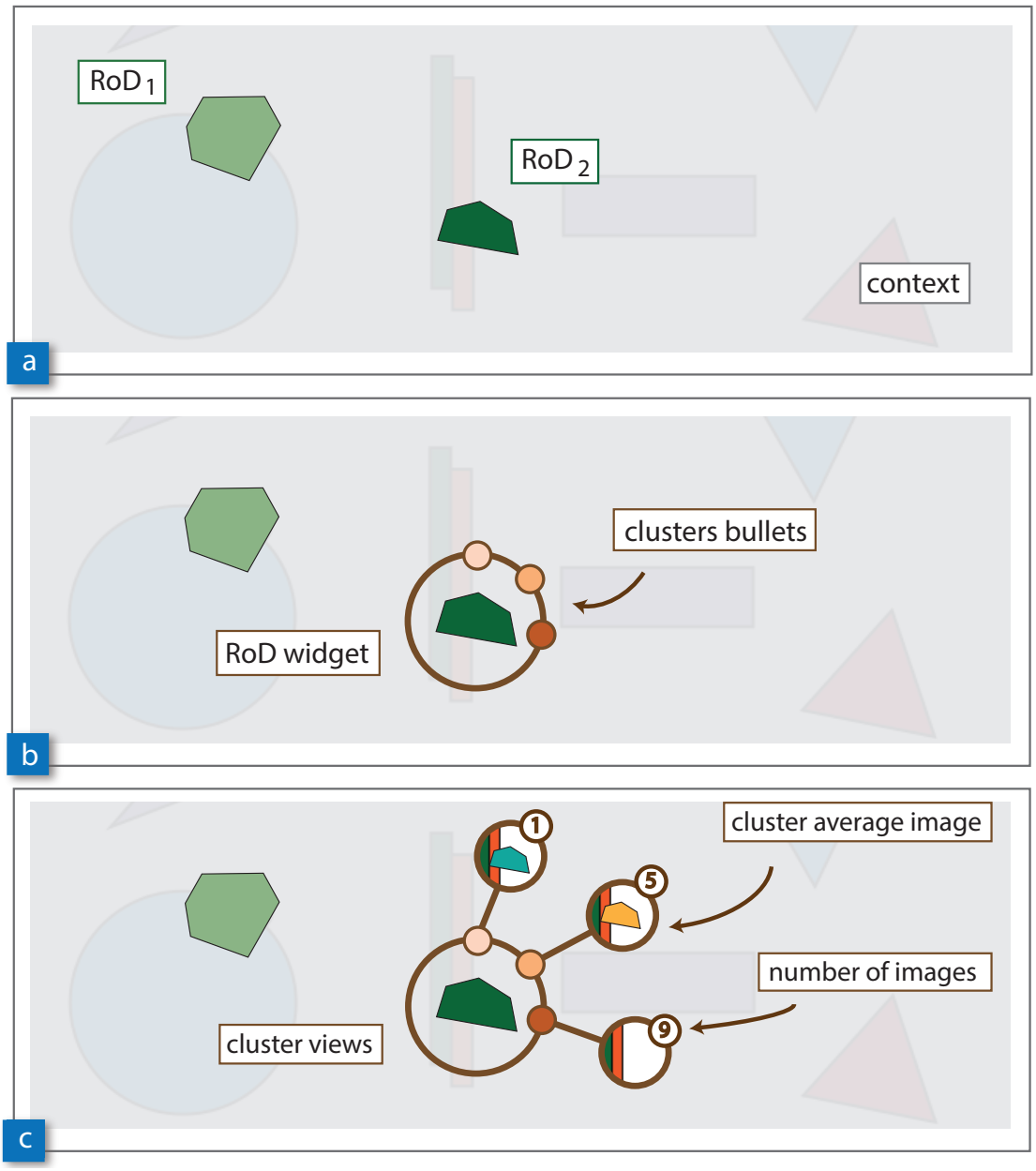

Figure 2.7: Main components of the interactive user interface. The image space is displayed in one view (a). Similar information is displayed in the background (*context*) and the variations in the images are highlighted by RoDs. The RoDs can be further explored individually by using the assigned *RoD widgets* (b). The *cluster bullets* assigned to the RoD widget depict the clusters available for the corresponding RoD. The bullets can be expanded (c) to *cluster views* which consist of the assigned *cluster average images* and icons depicting the *number of images* in the cluster. Such an icon can be used to retrieve the list of images assigned to the cluster.

be further analyzed by the users. Color-coding is used in the visualization to indicate a higher number of RoD-related clusters. A higher number of clusters is mapped to a darker color of the RoD polygon. Color-coding allows us to integrate the cluster information in the RoD visualization without occluding additional contextual information. In Figure 2.7a, the number of clusters for RoD_1 is lower than the number of clusters for RoD_2. Therefore, the color of RoD_2 is darker in the visualization.

RoD polygons can be inspected individually through mouse manipulation. When activated, RoD polygons are expanded to form **RoD widgets**. The widgets are displayed in image space in relation to the corresponding RoD polygons (Figure 2.7b). This embeds the widgets into the appropriate context and enables the users to analyze image variations without switching between different views. RoD widgets consist of a circle with colored bullets arranged around it. The colored bullets of the RoD widgets depict the corresponding clusters. The colors of the **cluster bullets** change according to the number of images that are included in the respective cluster. The color hue of the RoD polygon is always different from the hue of the bullets to prevent ambiguity. The color of a bullet is the darker, the more images the corresponding cluster contains. The bullets are ordered according to the number of images in the clusters. The users can decide whether they should be ordered in descending or ascending order. This allows users to decide whether the cluster with the highest number of images should be listed first (i.e., to quickly spot common patterns in the data), or whether the cluster with the lowest number of images should be on top (i.e., to find outliers in the data).

The cluster bullets of the RoD widgets can be further expanded by mouse manipulation (Figure 2.7c) to display the **cluster views**. Clusters are then represented by the average image which is derived from the images in the clusters. This immediately gives an overview on the image data that is encoded in the cluster. To get more quantitative information about the cluster size, the number of images is depicted for every cluster in a separate icon. These icons are attached to the average cluster images and provide additional functionality which can be controlled by mouse manipulation. When activated, the list of images encoded in the cluster can be viewed. The original input images are shown in a popup-window, and users can enlarge them by mouse interaction. This way users can very quickly identify groups of images in the original data, based on certain patterns.

In addition to the condensed view on the data, every RoD widget provides means to further analyze the results of the hierarchical clustering process. Mouse manipulation of the RoD polygons can also be used to view a visualization of the complete **clustering tree**. We use a dendogram approach, since this gives an instant structural overview of the hierarchical clustering results. A dendrogram is a tree diagram which illustrates the arrangement of clusters produced by hierarchical clustering. Clusters which are created by merging are represented by the average image computed from their leaves. Visualizing the clustering tree helps to understand how the final RoD-related clusters have been generated. Furthermore, the number of hierarchy levels in the tree indicates the diversity of the RoD-related image parts. The clustering which is visible in a RoD widget refers to a specific level in the clustering tree. The elbow criterion has been used to select the clustering (as described in Section 2.2.2). This decision may not be accurate in all cases. We therefore enable the users to overrule this decision by simply selecting another level in

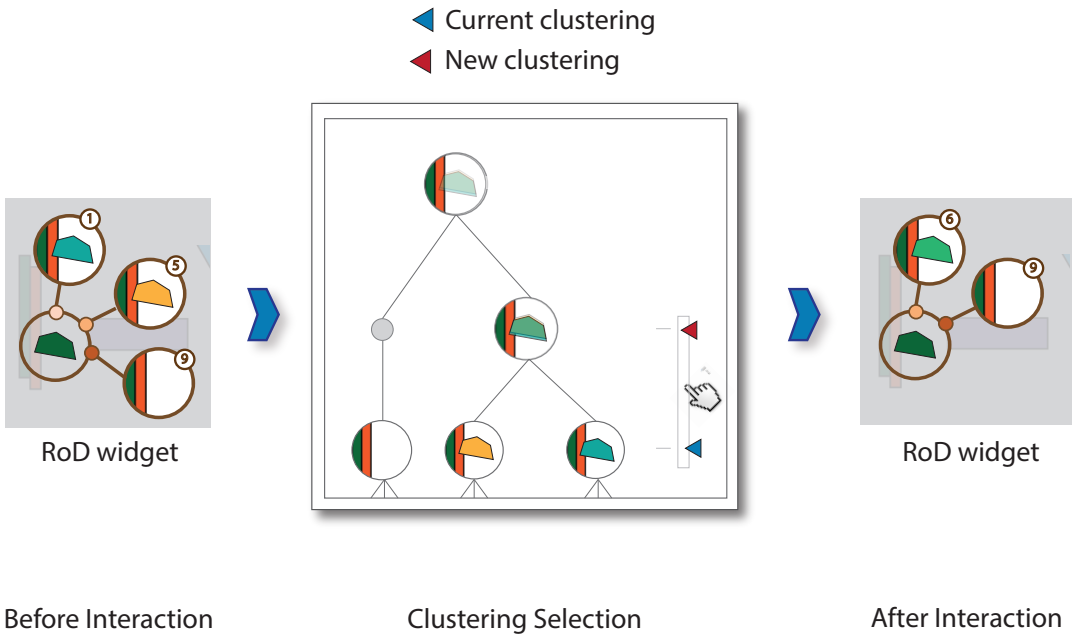

Figure 2.8: Interactive clustering tree. The default clustering for every RoD is selected automatically (*before interaction*). In the interactive clustering tree the users can specify a new level by mouse manipulation (*Clustering selection*). This will overrule the decision made during the clustering process. The new clustering is immediately available in the RoD widget (*after interaction*).

the *clustering selection*. The users can select a new level in the clustering tree by moving a slider. This process is illustrated in Figure 2.8. The corresponding RoD widget is immediately updated to represent the new clustering. Two leveles in the clustering tree are excluded from the selection, namely the top and the bottom level. The top level of the clustering tree contains only one cluster which includes all available images. At the bottom level of the clustering tree, all clusters only contain one image.

Additional **control tools** are provided in VAICo which users can employ to influence the visualization. The color of both the RoD polygons and the RoD widgets can be changed according to given color schemes. To skip variations that are not of interest for the data analysis, and to prevent the visualization from getting overly cluttered, users can employ command tools to hide individual RoDs. By default, RoD widgets are drawn around the corresponding RoD polygon. According to the size and shape of the image variations, it may happen that RoD widgets cover a large area of the image space when activated. Therefore, the RoD widgets can be shrunk. In the case of a biased image comparison, the semantics of the color coding of the RoD polygons are slightly different, since information is collected on the number of images the RoDs are present in. The darker the RoD polygon, the higher the number of corresponding image parts. This way the colors of the polygons depict outliers in the data that are either available in many, or in just a fraction of the input images.

2.3 Implementation

The raw data for the presented approach comprise an ensemble of unsorted PNG images. The pre-processing, consisting of an image comparison step and a data clustering step, has been implemented in JAVA. The interactive visualization tools have been implemented as an interactive web application. The widgets are embedded in an HTML5 canvas and interactions are done in JavaScript (using the library KineticJS[1]).

The cost of the pre-processing depends on the number and size of the images in the set. The run-time depends on the number of identified RoDs, since for every RoD hierarchical clustering has to be applied. The pre-processing for the largest data set called *Gene Expression*, which consists of 578 images of size 200x200 pixels, takes 2.1 minutes. Afterwards the results of the pre-processing step are transferred to the visualization system by JSON files. The results are then loaded into the visualization system, which means that the provided JSON files are parsed. This takes 10 seconds for the largest data set. Afterwards the visualization system is ready to be utilized by the users. The interaction itself works in real-time.

2.4 Results

In order to evaluate the proposed visualization technique different ensembles of images have been analyzed. The images are taken from different domains to show the applicability of the proposed method to different types of datasets. An overview of the image ensembles is given in Figure 2.9.

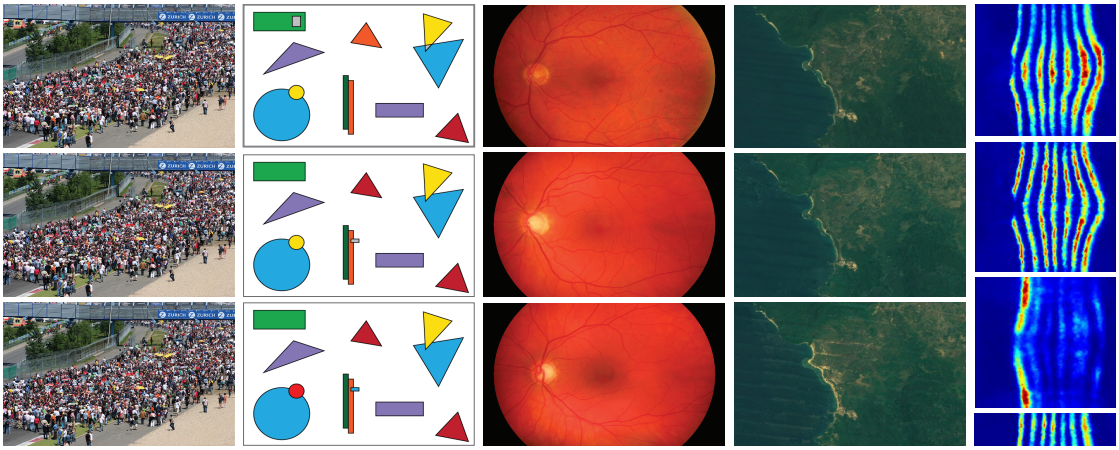

Figure 2.9: Image datasets. The dataset *Puzzle* contains pictures of a real-world scene (*first column*). The dataset *Shapes* contains images with shapes of different colors (*second column*). The dataset *Retina* contains retina images from different patients (*third column*). The dataset *Satellite* comprises satellite images of a coast-line in Indonesia (*fourth column*). The dataset *Gene Expression* contains images with color coded gene expression information (*fifth column*).

[1] http://kineticjs.com/

The selection of the dataset **Puzzle** was inspired by the well-known spot-the-difference puzzles. This dataset consists of ten pictures which show a real-world scene. The images are of size 1050x700 pixels. In the images small elements disappear or change their color. We used the unbiased (i.e., pixel-based) image comparison approach to locate the image differences. VAICo then takes the results of the comparison to give an instant overview of all variations in the datasets. The RoD widgets can be used to reveal the underlying raw data that actually caused the difference in the data. The VAICo visualization of the *Puzzle* dataset is depicted in Figure 2.10. It can be seen that four RoDs have been identified that contain two main clusters each. These RoDs correspond to features that are present in some of the images, and are missing in other images. For example, in the right lower corner, there is a men visible in seven of the images, but not in three. One of the RoDs contains three main clusters. Here an umbrella is colored differently in the images, which can be clearly seen when opening the RoD widget's cluster bullets.

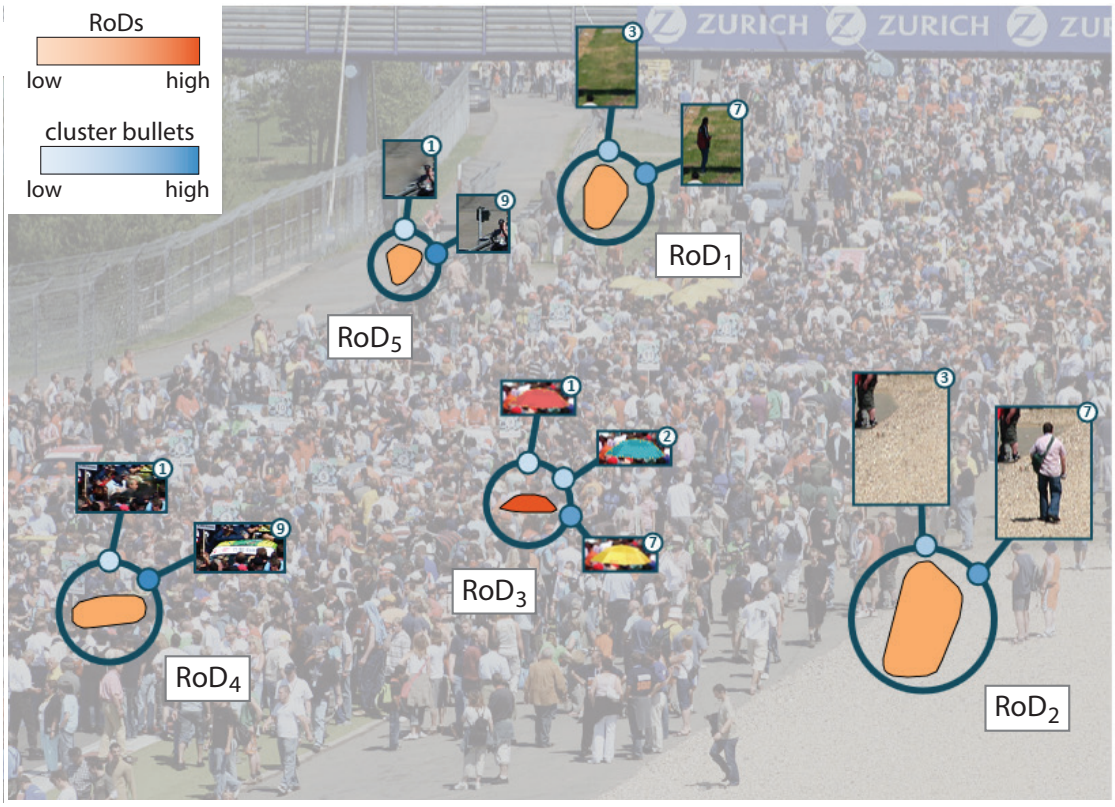

Figure 2.10: Results for dataset *Puzzle*. Our approach identified five RoDs which depict the data changes in the images. One object in the scene changes its color (RoD_3), and some other objects are not present in all images (RoD_1, RoD_2, RoD_4, RoD_5). The color-coding of the RoDs shows the number of corresponding clusters. The color-coding of the cluster bullets indicates the number of assigned images.

The dataset **Shapes** is a synthetic dataset which consists of 52 images. The images are of size 600x400 pixels. They contain different types of shapes (rectangles, triangles and circles) of various colors and sizes on a white background. In the series of images, individual shapes disappear, re-appear or change their color. We used the unbiased (i.e., pixel-based) image comparison approach to locate the image differences. An overview of the VAICo visualization of the dataset can be seen in Figure 2.11. The RoD widgets of three of the available four RoDs have been expanded in the Figure to see the underlying details. Similar to the *Puzzle* dataset, images for which certain shapes are missing, are clearly visible in the visualization. Also another case can be depicted when examining the RoDs. For the RoD in the middle, it can be seen that a rectangle at that location changes the color, as well as that it is also missing in some of the images. The sizes of the clusters also indicate how many images are effected, and how the differences are distributed among the images.

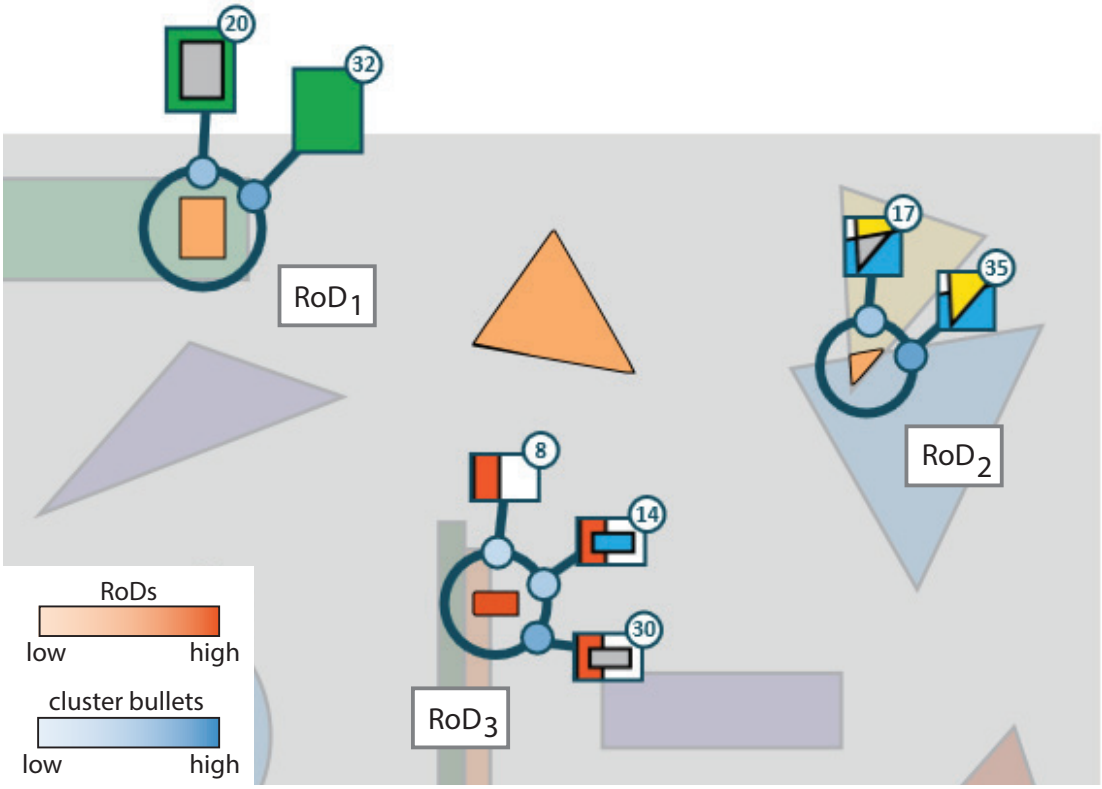

Figure 2.11: Results for dataset *Shapes*. This dataset consists of images containing shapes of different colors on a white background. Our approach identified four RoDs (three of them expanded in this Figure). Some of the objects are not present in all images (RoD_1, RoD_2), and some of them also change their color (RoD_3). The color-coding of the RoDs shows the number of corresponding clusters. The color-coding of the cluster bullets indicates the number of assigned images.

The dataset **Retina** contains 20 retina images of different patients. The images are of size 1168x779 pixels. In some of the images anomalies are present, because the patients suffered from diabetic retinopathy. Retinopathy is a damage to the retina caused by complications from diabetes. Blood vessels at the back of the eye can bleed and blur vision. The leakage of blood can be seen as dark spots in the retina images. Through these dark spots, retina images of sick patients can be differentiated from retina images of healthy patients. We also used the unbiased image comparison approach to analyze this dataset. The results can be seen in Figure 2.12. In the comparative visualization of VAICo, the dark spots which were present in some of the images resulted in the creation of RoDs at these positions. When investigating the RoDs, these images containing dark spots on the retina can be identified as outliers in the data. Images without anomalies are merged into one cluster since they contain similar information. The RoD widget allows us to explore the variations in the data, so that anomalies can be distinguished from normal data variations that do not indicate pathological changes. The RoD widgets provide the possibility to retrieve the original image data, which keeps track of the investigated patients.

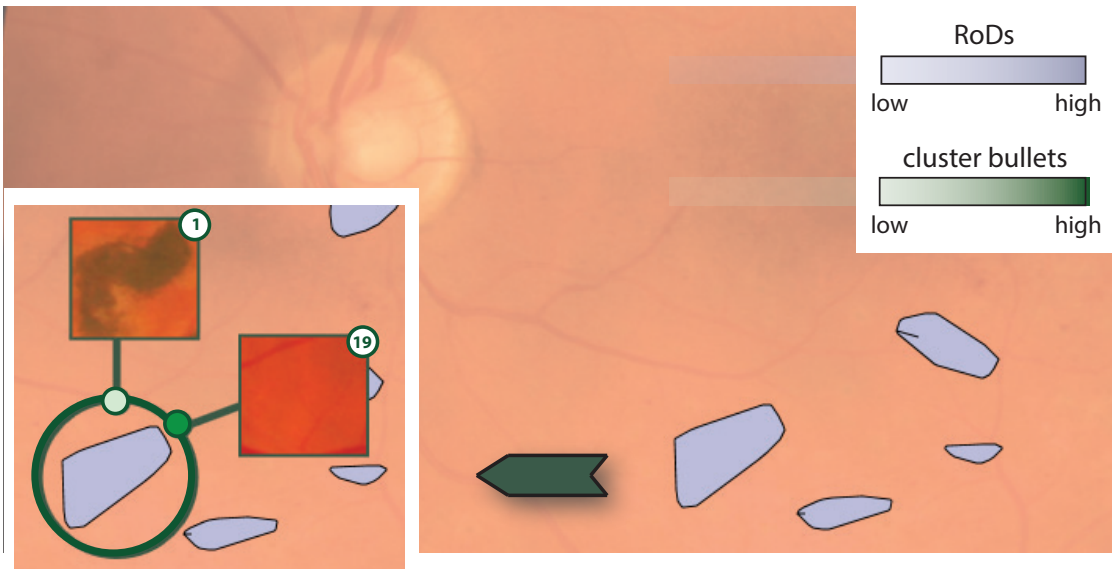

Figure 2.12: Results for dataset *Retina*. This dataset consists of retina images from different patients. The image comparison identified dark spots on the retina, which can be further analyzed by using the RoD widgets. The color-coding of the RoDs shows the number of corresponding clusters. The color-coding of the cluster bullets indicates the number of assigned images.

The dataset **Satellite** consists of 12 satellite images of size 614x450 pixels. The images show a coast-line in Indonesia and cover a time period from 2000 to 2011. One image has been produced every year. The coast-line has been greatly affected by a tsunami in 2004. We used the unbiased (i.e., pixel-based) image comparison approach to analyze this dataset, since no prior information

was given about the structure of the image data. It was necessary to adjust the threshold to skip small variations in some parts of the data. When viewing the comparison results with VAICo, the image part showing the coast-line is covered by one RoD. The results of the visualization can be seen in Figure 2.13. VAICo allowed to detect very interesting patterns in the data. As it can be seen in the Figure, the image showing the damage caused by the tsunami on the coast-line is clearly visible as an outlier in the RoD data. The image is contained in one separate cluster (leftmost cluster in Figure 2.13). The In the remaining two clusters, the images before (middle cluster) and after (rightmost cluster) the tsunami are summarized. When manually inspecting the images, such patterns and outliers that are present in the data could be easily missed. Additionally, differences between the state of the coast-line before and after the tsunami are not inherently available in the dataset.

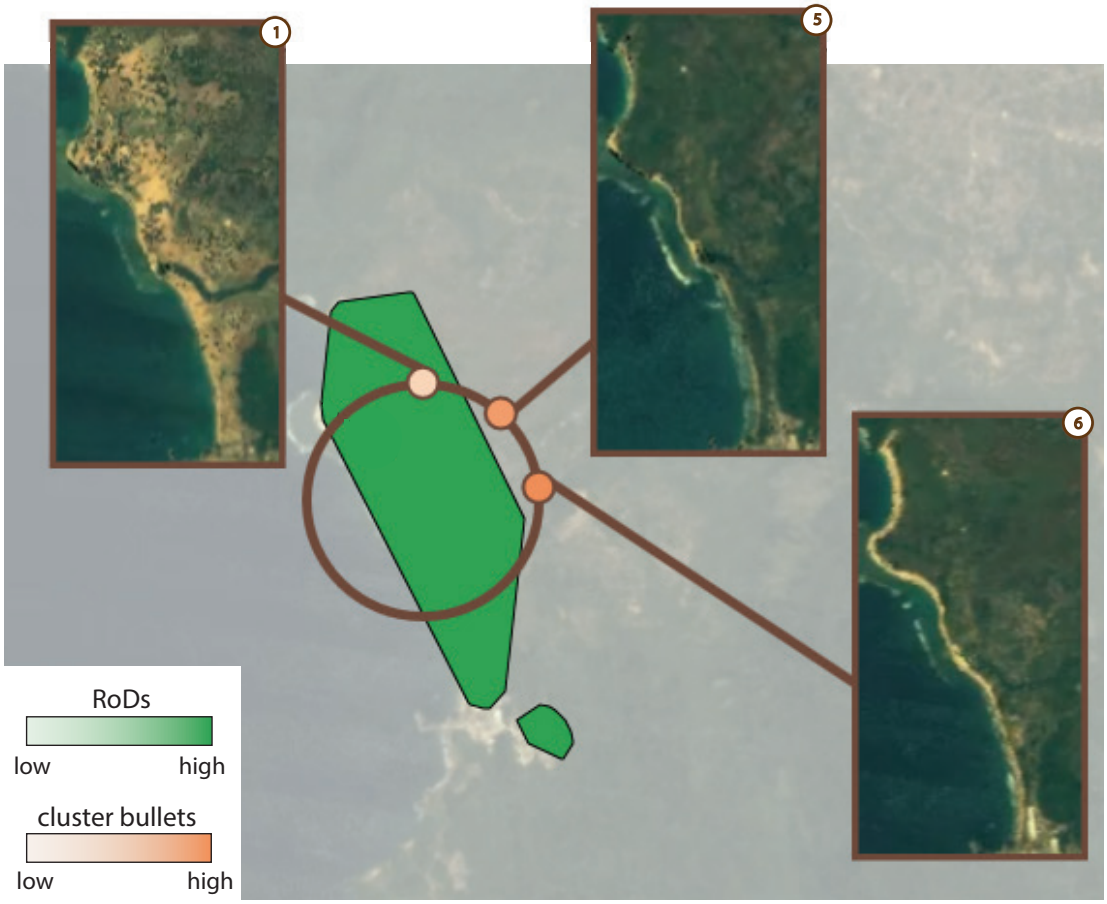

Figure 2.13: Results for dataset *Satellite*. The images in this dataset show a coast-line in Indonesia. Damage caused by the tsunami in 2004 is identified as an outlier in the data (*leftmost cluster*). The images showing the state of the coast-line before (*middle cluster*) and afterwards (*rightmost cluster*) are summarized in separately. The color-coding of the RoDs shows the number of corresponding clusters. The color-coding of the cluster bullets indicates the number of assigned images.

The dataset **Gene Expression** originates from the biological domain and consists of 578 images showing gene expression data. The images are of size 200x200 pixels. The data has been created from point cloud datasets of different fruit fly embryos as described by Fowlkes et al. [FHK+08]. In the images gene expressions of the *EVE* protein are color-coded from very high (red) to very low (blue). Every image in the Gene Expression dataset corresponds to one fruit fly embryo. We analyzed this dataset by using the biased image comparison approach. We located the seven stripes of the *EVE* protein in the images by segmentation. Then the RoD data of the images was compared and the results were visualized with VAICo (Figure 2.14). The color coding of the RoD polygons indicates the number of images the RoDs are present in. In the visualization it can be seen that the first stripe shows up in more images than the other six stripes. This is due to the fact that some images covering the earlier stage of the embryo development only contain one stripe. Additionally, outliers can be detected very easily in VAICo. They refer to patterns in the images where the gene expression does not look like as expected.

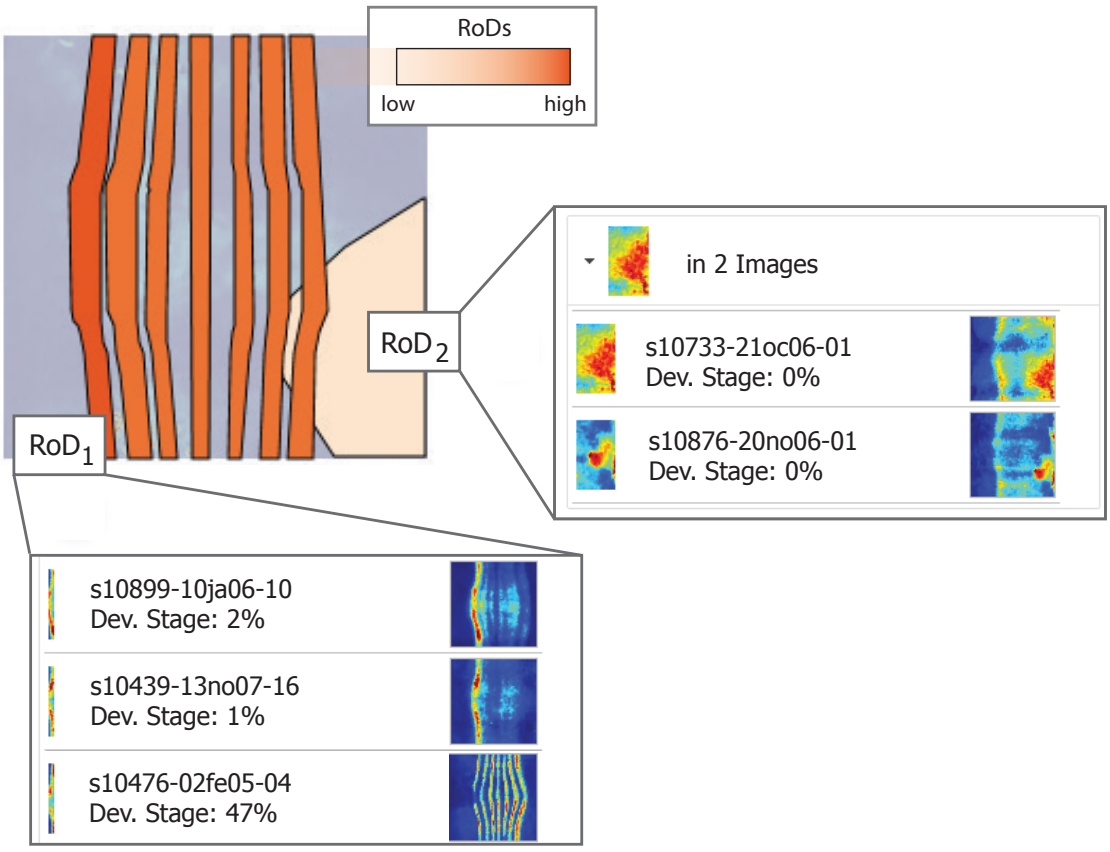

Figure 2.14: Results for dataset *Gene Expression*. The color-coding of the RoD widgets depicts in how many images the corresponding RoD can be found. The first stripe (RoD_1) is available in more images than other stripes. An outlier region RoD_2 could also be identified, which shows unexpected gene expression data (available only in 2 images).

2.5 Evaluation

We collected user feedback to evaluate the presented visualization techniques. We hypothesized that using our visualization technique, participants would: (1) get results faster when searching for differences in a set of images; (2) get a better overview of individual variations in the image data; and (3) be able to better spatially localize differences in the image data.

For the feedback we created a **setting** were we could compare a juxtaposition of images to the comparative visualization VAICo as presented in this chapter. In the juxtaposition participants could scroll through the list of images, sort them by drag-and-drop and click on images to enlarge them. VAICo, on the other hand, contained all the main features, like that the ensemble is presented in one view, and that users could use the RoD widgets to explore the details. An overview of the evaluation setting is given in Figure 2.15. The images taken from the *Shapes* dataset contained shapes like triangles, circles and rectangles on a white background. Some of the shapes changed their color or disappeared in the image set. For the evaluation, six different dataset versions have been prepared. Due to their very general composition, images could be easily interpreted by users from different domains.

We designed **three tasks** which refer to the three hypotheses mentioned above:

- *T1: Depict variations with certain parameters*. In this task participants had to identify one shape which is present in all images. None of the remaining shapes did show up in all of the images.
- *T2: Analyzing local variations*. In this task participants had to identify one shape that is colored in three different ways in the images. Then they had to sort the three colors for this shape by the number of occurrences.
- *T3: Localization of changes*. In this task participants had to identify the most mutable image region (i.e., the image space part with the highest number of variations).

Participants had to complete all three tasks twice: once by using the juxtaposition and once by using the VAICo visualization system. Datasets were switched between different tasks, and also between different visualization techniques. Therefore, participants never worked on the same dataset twice. The order of the tasks (as stated above) was the same for every participant. The order in which the two different visualization techniques were presented to the user was selected at random. For every task we measured the completion times and the correctness of the answers. The user study was conducted with 11 participants from three different domains, namely computer science (7), engineering (3), and medicine (1). Participants were between 26 and 57 years old. The group of participants consisted of 9 men and 2 women.

The **results** of the user study are presented in Figure 2.16. The results show that VAICo definitely improves the search time for image differences. In addition to time, the error rate was measured for every task. For the juxtaposition some participants made errors when trying to solve task T2 (error rate of 9.1 %) and task T3 (error rate of 27.3%). Participants did not give wrong answers when using the VAICo system. When analyzing the results, it can be seen that VAICo helps users

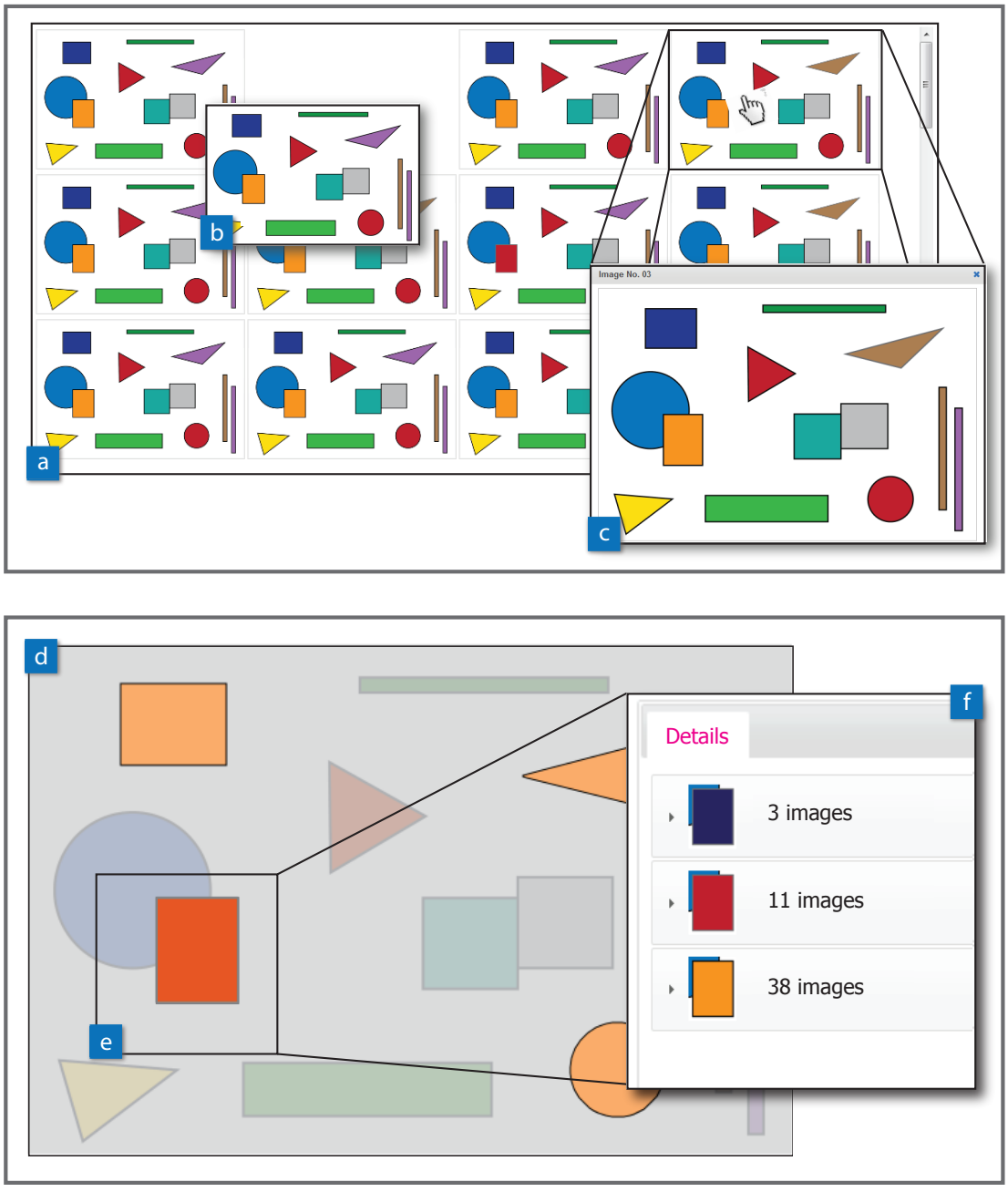

Figure 2.15: Evaluation setting. The users had to solve the same tasks in a juxtaposition (upper part), and in our proposed VAICo system (lower part). In the juxtaposition, images were placed side-by-side (a). Users could scroll through the list of images, sort images by drag-and-drop (b), and check individual images by clicking on them (c). In the VAICo system, the image ensemble was presented in one view (d). Users could employ the RoD widgets (e) to get more details about individual image differences (f).

to quickly get an overview on differences in the image set (task T1). It also helps to analyze local variations (task T2). Both tasks would require time-consuming analysis steps if done manually by the user. The comparative visualization brings significant improvements for users to localize variations in image space (task T3). This is a very tedious task if done manually.

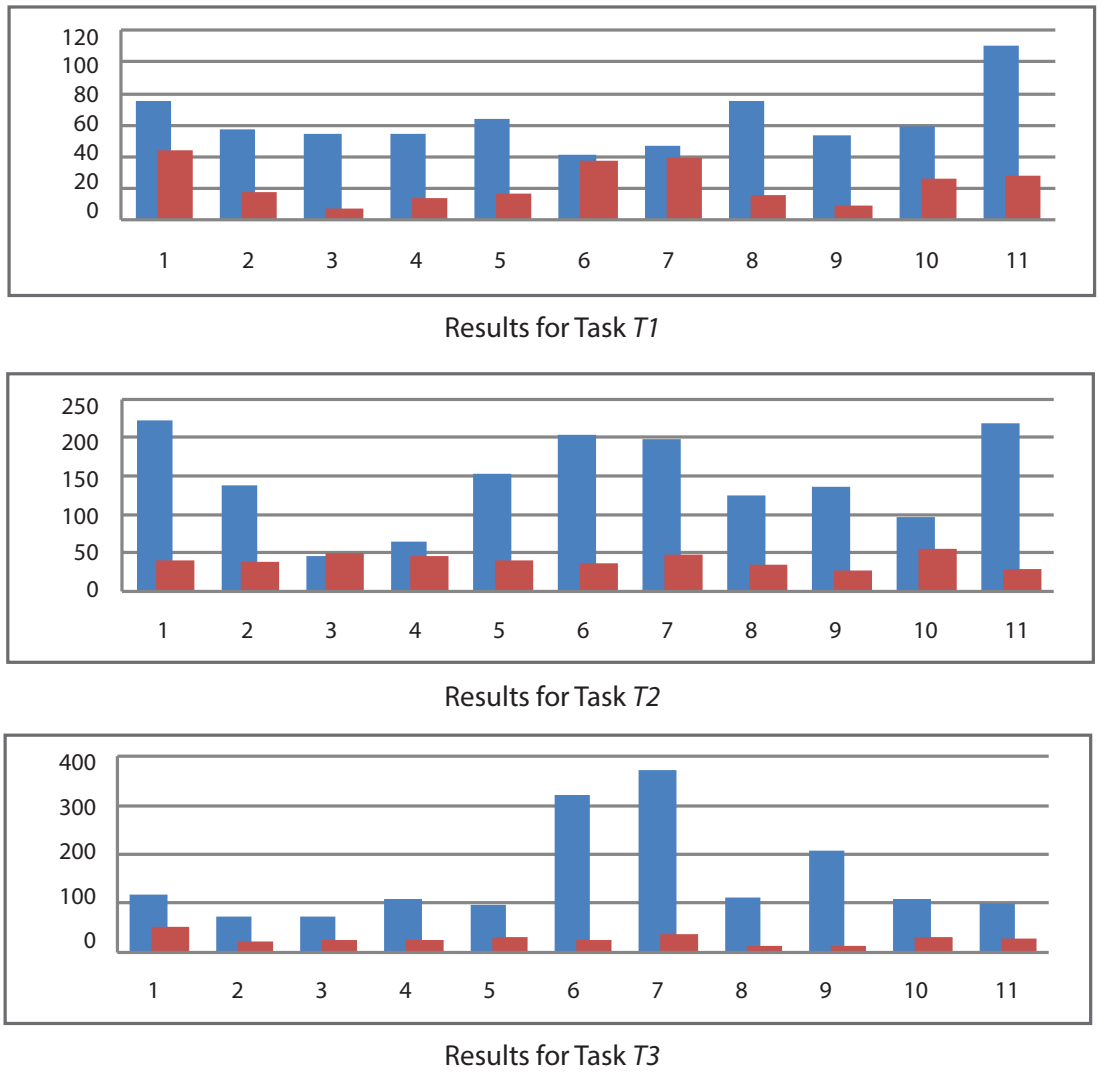

Figure 2.16: Evaluation Results. The charts indicate the time it took the participants to complete the tasks T1-3 with the juxtaposition (blue) and with VAICo (red). The x-axis depicts the participant number (1-11) and the y-axis gives the time (in seconds) it took to complete the task. Every participant had to complete every task twice: once by using a juxtaposition and once by using VAICo. Therefore, two different measurements are available per task for every participant. The evaluation results show that VAICo clearly is of benefit for completing the examined tasks.

2.6 Summary

VAICo is a visualization technique for the comparative visualization of image ensembles. Interactive visualization tools are provided to explore the image space and drill-down on individual variances. Our visualization approach addresses the scalability of image comparisons and proposes ways to integrate contextual information and more detailed information in one view. Contextual information is preserved, whereas image variances can be efficiently spotted and put into context. Our approach can be applied to quickly identify small local differences in an image set. It is also helpful for analyzing the occurrence of previously defined image features.

VAICo is scalable to a large number of images. We use clustering to cope with scale and to identify patterns in the data. Clusters are depicted as bullets around the RoD widgets. If many clusters have to be displayed in the widget, there might not be enough space around the widget to visualize all of them. In this case additional functionality is provided to the users, which means that clusters are presented in a separate popup-window (where scrolling can be activated easily). However, we would like to point out that a **high number of cluster bullets** for a RoD usually indicates that the underlying clustering should be refined. VAICo provides the possibility to adjust the clustering by selecting another level in the clustering tree. We think that having the ability to refine the clustering is more appropriate for analyzing the data, than providing means to deal with high numbers of clusters. Including additional visualization techniques which can handle these special cases might be an interesting idea for future work.

The used *Mean Squared Error* (MSE) computation is a very simple image comparison approach which has some limitations. As one approach to deal with these limitations, we provide the possibility to adjust the sensitivity of the algorithm. The users can define a **sensitivity threshold** for intensity changes. Figure 2.17 illustrates how the threshold effects the RoD calculation. The higher the threshold, the less pixels will be considered as variations in the data. Increasing the threshold will shrink the resulting RoDs, and might even cause RoDs to disappear. This way, particular intensity variations in the data can be skipped if they are not of interest for the data analysis. MSE is very sensitive to global intensity shifts, and adjusting the sensitivity threshold might not be enough to deal with global changes in the data.

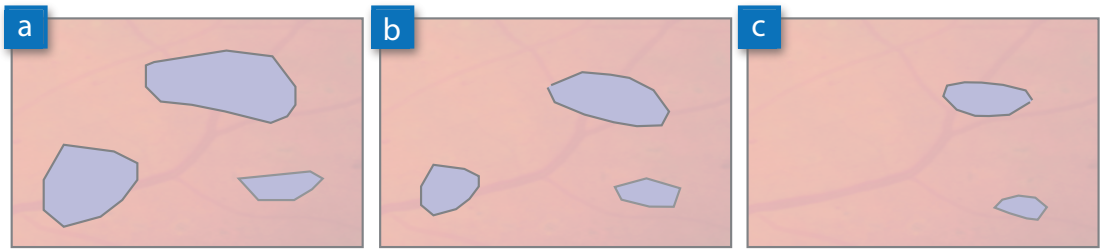

Figure 2.17: Effect of sensitivity threshold. Three different threshold have been applied, increasing values from (a) to (c). A higher threshold values results in less image space being occupied by the regions of differences (RoDs).

VAICo operates on a set of images, where the individual images exhibit **small localized differences**. The approach has its limitations if dealing with a high number of variations in the data. Figure 2.18 shows two examples where VAICo does not produce appropriate results, namely varying data in landscape images and motion data. VAICo is designed for scenarios where the obtained images have undergone a certain form of standardization. This is often done during the acquisition process itself (e.g., by employing a particular protocol for mounting and imaging) or through post-processing such as registration. As such procedures are highly dependent on the application domain, they are not the focus of our work. Instead, we assume that all images in our input set have undergone such a process, i.e., they are of the same size and represent similar regions in space. In the case of landscape data (Figure 2.18a), an additional clustering step could be applied to the images, and VAICo could be used to further explore images in the individual clusters. In the case of motion data (Figure 2.18b), VAICo gives a quick overview on the motion tracks in the images. Although our approach is currently not suited to deal with time-dependent data, it will be interesting to explore its applicability for event detection in video data.

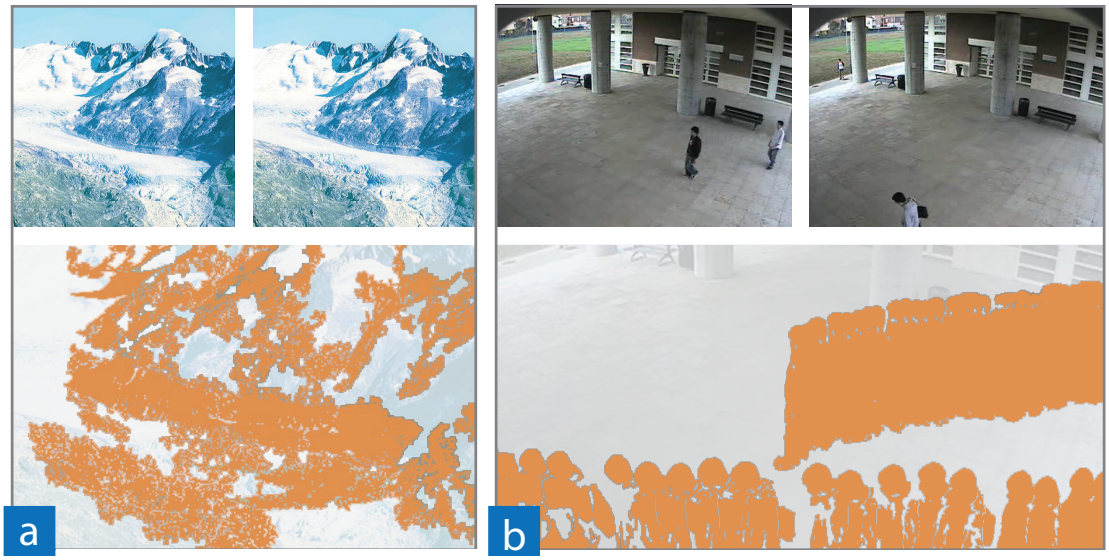

Figure 2.18: Data analysis limitations. Landscape images (a), that include a high number or variations, and motion data (b) result in very large RoDs. This is because VAICo was designed for images that exhibit small local differences. Adjusting VAICo for other types of images is an interesting topic for future work.

CHAPTER 3

Visual Analysis of Differences in Mesh Ensembles

This chapter is based on the following publication:

Johanna Schmidt, Reinhold Preiner, Thomas Auzinger, Michael Wimmer, M. Eduard Gröller and Stefan Bruckner. YMCA – Your Mesh Comparison Application. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, VAST '14, Paris, France, pp. 153-162, November, 2014

Polygonal meshes are one of the most commonly used surface representations in 3D computer graphics. Their explicit description of the surface location in 3D together with local connectivity information enables memory-efficient storage and provides a convenient data structure for a wide range of applications (e.g., in geometric processing). For many tasks related to mesh creation and/or editing, a multitude of proposed methods exist. Polygonal meshes may serve both as input and output for a majority of such techniques. As a consequence, the characteristics and capabilities of different approaches for a common task have to be evaluated on the basis of their results, which inevitably leads to the need to compare an ensemble of similar meshes. The analysis of such mesh ensembles is of major importance for several geometric processing tasks such as mesh resampling, mesh simplification, and mesh denoising. Beyond computer graphics, other fields that deal with 3D objects, like CAD or biomolecular modeling, also benefit from new trends for multi-mesh comparison. The analysis of differences in 3D poses several interesting challenges. Firstly, if the differences should be explicitly encoded, a proper metric for comparing the 3D data has to be found. The metric also strongly depends on the analysis tasks that have to be solved. Secondly, for the encoding and exploration of the differences in a 3D environment, it has to be ensured that interesting regions are not missed by the users due to occlusions. Until the publication of this work, the tool-set for the visual comparisons of 3D meshes was limited to basic techniques for comparative visualization. Analysts typically employ statistical evaluation (e.g.,

computing the global error), simple juxtaposition, or explicit encoding by color to visualize mesh differences. This is illustrated in Figure 3.1. As mentioned in Section 1.2.2, juxtaposition does not scale well with the number of instances, and basically supports only pairwise comparisons. If the visualization of the differences cannot be summarized in one view, it is necessary to scan/rotate/zoom into several 3D meshes one after another, and manually detect and remember all small differences in the data. Explicit encoding (e.g., color-coding of differences) solved this problem by pointing users to the regions in the data where differences occur. The technique, however, only partially characterizes the behavior of the underlying algorithm (e.g., whether the data is smoothed). Similar to this, a global statistical evaluation of the meshes in the ensemble might not reveal all visual properties like small details of the input meshes.

As an exemplary use case for 3D mesh comparison, we concentrated on the analysis of meshes created by different *mesh reconstruction* algorithms. Mesh reconstruction refers to extracting meshes from point clouds as accurately as possible [BLN+13]. Point clouds represent the external surfaces of objects, and are typically created by 3D laser scanners. The advent of affordable scanners has made the creation of virtual representations of real-world objects a commodity. Point clouds are generally not directly usable by 3D applications, and are therefore usually converted to polygonal or triangle meshes. A wide variety of mesh reconstruction techniques has already been developed, and these algorithms differ (more or less) subtly in their reconstruction behavior. Especially the presence of noise, outliers or other errors in the input data may influence the result of the reconstruction [BTS+14]. Furthermore, with almost every technique the output depends

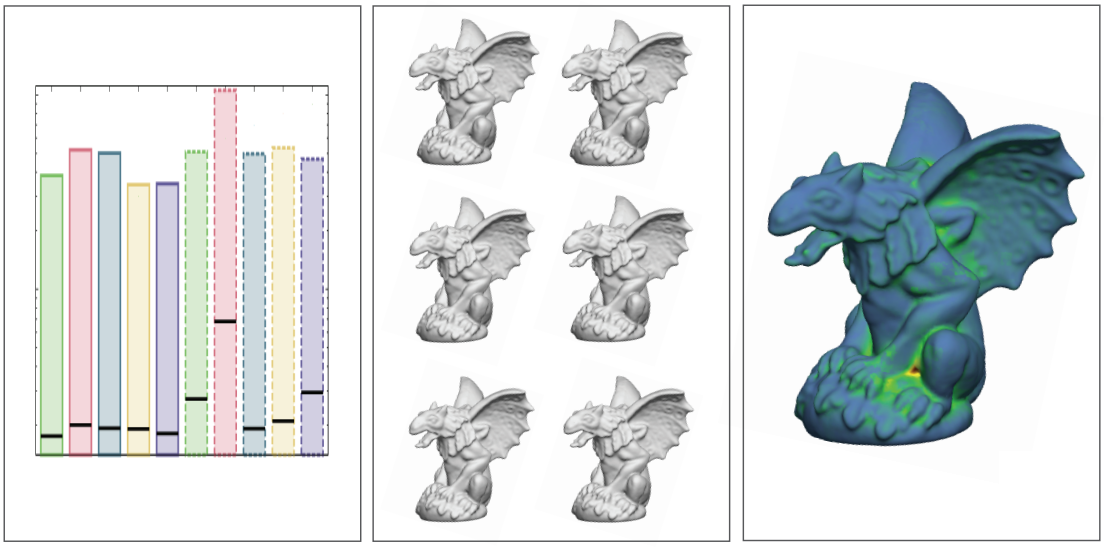

Figure 3.1: Common mesh comparison approaches. Current tools often employ statistical evaluation (*top*) to visualize the quality of reconstructed meshes, which typically does not allow to visually judge the mesh differences. This problem can be solved by employing juxtaposition (*middle*). Another possibility is to use explicit encoding by color to show differences between meshes (*bottom*).

on several, partly very sensitive, parameters with varying suitability for different kinds of data. All these facts create a large space of possible results when reconstructing a mesh from a point cloud. From a statistical point of view, the quality of a reconstruction can be defined by the residual distances of each surface point from the reference shape. As mentioned before, such a global evaluation alone hardly communicates a full understanding of a technique's strengths and weaknesses on different types of data. In practice, users often face a trade-off between the preservation of geometric detail and the robust removal of scanning artifacts like noise or holes. It is therefore required to keep the human in the loop and let the users judge upon aesthetic considerations during the analysis. Visualization systems need to provide means that allow users to inspect the visual properties of the input meshes in a 3D environment. When implementing mesh reconstruction techniques, a special challenge is the evaluation of a newly developed algorithm, as until now it required extensive laborious comparisons. It is necessary to compare samples of the approach in its own parameter space, as well as to compare the new technique with existing state-of-the-art methods.

In this chapter we describe our findings for the visual analysis of 3D mesh ensembles. We propose a combination of explicit encoding, juxtaposition and quantitative measures that supports mesh comparison tasks and provides more insight into the underlying data. We implemented an application targeted towards the comparative visual analysis for 3D mesh ensembles. Our application is called YMCA - **Y**our **M**esh **C**omparison **A**pplication. YMCA automatically compares the meshes in the ensemble and helps to identify areas in the data where reconstruction algorithms produce differing results. Our application further allows for a detailed exploration of local regions of interest, so that users can visually judge the characteristics of different mesh reconstruction algorithms. The different results of the reconstructions in the local regions of interest can also be compared with each other.

The main features of our approach are:

- *Comparison of ensembles*: Our visual analysis methods are designed to overcome the problems of previous approaches that do not scale well with mesh ensembles. With our approach users are able to get an overview of all studied algorithm results. It is also possible to evaluate the performance of individual algorithms with others.
- *Focus+Context*: As a starting point of the analysis process, we provide an overview of the comparison results. Users can then further concentrate on local variations and explore them in more detail without losing the context information.
- *Flexibility*: The proposed visual analysis tools can be applied to different mesh comparison tasks, e.g., comparing meshes after mesh simplification, as well as comparing different reconstructed meshes. The approach is neither tied to a certain type of mesh (e.g., watertight mesh), nor to a certain mesh comparison metric.

3.1 Related Work

YMCA is related to comparative visualization (as already introduced in Section 1.2) and ensemble visualization (as already mentioned in Section 1.2.2). Our application further addresses the problem of **multi-mesh comparison**. Due to the need to evaluate mesh editing techniques (e.g., mesh simplification or mesh denoising), many approaches have been developed that support multi-mesh comparison. Various techniques focused on the mathematical background and established metrics which can be used to compare meshes. Aspert et al. [ASCE02] proposed an approach to measure differences between two meshes by using the Hausdorff distance. Roy et al. [RFT04] introduced a new mesh comparison method using an attribute deviation metric. *MeshLab*, by Cignoni et al. [CCR08], was implemented to combine mesh comparison as well as mesh editing tools. In our work we focus on visual support for mesh comparison, and some interesting approaches have already been developed in this area. Cignoni et al. [CRS98] presented *Metro*, a system that allows for pairwise comparison of surfaces. A similar approach was later proposed by Silva et al. [SMS05]. Their system, which is called *PolyMeCo*, allowed users to compare meshes with a reference mesh. Existing approaches for mesh comparison use color to encode the differences and present the results by juxtaposition. Therefore, they are limited to a small number of meshes. Apart from zooming, the systems also do not provide means to inspect local areas. In our approach we extend these ideas to provide means to compare multiple meshes, and to inspect local regions in more detail.

The acquisition of virtual representations of scanned real-world objects from point clouds is referred to as **surface reconstruction**. In contrast to point-set surface-representations [AK04], this work focuses on mesh reconstruction from point clouds. Meshes are reconstructed according to different formulations of implicit surfaces defined on the input points, ranging from locally fitted tangent planes [HDD+92], radial basis functions [CBC+01], to Poisson reconstruction [KBH06]. All these techniques exhibit their own characteristic reconstruction behavior in terms of robustness and accuracy, and require various parameters which influence the result. Berger et al. [BLN+13] present a benchmark tool for surface reconstruction algorithms, where the users can test different algorithms on different point cloud datasets. When presenting the results, they use juxtaposition where rendered models are placed side by side. This way the complex task of finding relevant differences in the data is shifted to the user.

The concept of **linking-and-brushing** is well-known in visualization. It refers to the connection of two or more views in a way that a change to the representation in one view affects the representation in the other one as well [War09]. Linking-and-brushing is a very flexible concept that can be applied to many different data representations, like 2D data (e.g., scatter plots [BC87]) as well as 3D data [DH02]. We use linking-and-brushing to keep track of users' selections. Elements in our summary representations can be selected, which will mark them as selected also in the detailed view (and vice versa).

3.2 YMCA – Your Mesh Comparison Application

YMCA combines explicit encoding, juxtaposition, parallel coordinates, and interaction techniques (i.e., linking-and-brushing and focus+context) to convey an overview of mesh differences, and to allow the users to inspect local areas of interest. We focus on triangular mesh data produced by different mesh reconstruction algorithms. The data has been created by the surface reconstruction benchmark tool implemented by Berger et al. [BLN+13]. The meshes are already registered. No pre-processing (e.g., filtering) has been applied to the meshes. In addition, a reference mesh is assumed to be available for every point cloud.

To provide a condensed representation of all differences in the data, we propose to project the variances of the mesh deviations onto the reference mesh. The calculation of the variances is described in Section 3.2.1. In addition, we locate problematic regions (i.e., regions of high variance) in the model to provide additional guidance to the users when exploring the data. The detection of such regions is outlined in Section 3.2.2. The problematic regions are used to build a parallel coordinates plot to visualize the performance of the reconstruction algorithms (Section 3.2.3). The inspection of local areas provides interesting insights into the behavior of the reconstruction algorithms. The visual analysis tools of YMCA are described in Section 3.2.4. The whole pipeline of YMCA is outlined in Figure 3.2.

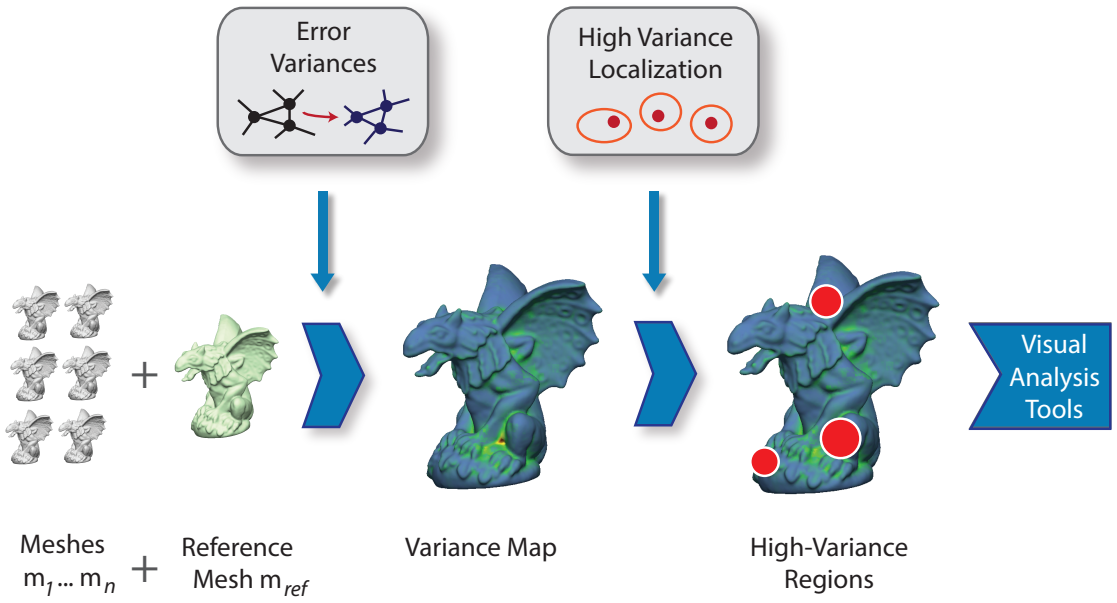

Figure 3.2: Overview of our visual analysis approach. The input data consists of a set of n *meshes* $m_1 \ldots m_n$ and one *reference mesh* m_{ref}. The surface deviations of the meshes are calculated to get the corresponding *variance map*. Afterwards *high-variance regions* are located in the data. The results are finally presented using our *visual analysis tools*.

3.2.1 Calculation of Error Variances

The usage of explicit encoding of the mesh differences requires the computation of the mesh differences in the ensemble first. For the *mesh comparison* we use an attribute deviation metric as described by Roy et al. [RFT04]. This metric can be employed to compare meshes in a pairwise manner. The metric calculates point-wise *deviations* from a first mesh to a second mesh. The deviation is defined as the distance between a vertex of the first mesh and the nearest point on the surface of the second mesh. In our case the first mesh is always defined to be the reference mesh. The reference mesh of m_{ref} is then compared to all other meshes in the ensemble in a pair-wise manner. In all cases the surface deviation values for all vertices of m_{ref} are computed. The surface deviation values can be denoted by *errors* in the following, because they describe differences between the reference mesh and the input meshes. Given n input meshes $m_1, \ldots, m_n$, the mesh comparison then results in a set of n error values for every vertex in m_{ref}.

We then calculate the per-vertex error variances for all vertices v in m_{ref}. The error variances are used to detect regions where algorithms differ in the way they handle noise or missing data. Some reconstructions produce holes in areas where other reconstruction algorithms are able to close the mesh. Some algorithms take outlier points into account which lead to bumps on the surface, which are smoothed away by other algorithms. Such cases can be identified in the data when using the error variances. First we need to compute the per-vertex mean error $mean_v$. Then the per-vertex variance σ_v^2 can be calculated based on the n error values $d_1 \ldots d_n$ with the standard formula:

$$\sigma_v^2 = \frac{\sum(mean_v - d_n)^2}{n} \tag{3.1}$$

These computations result in per-vertex variance values σ_v^2 for all vertices in m_{ref}. In YMCA, we call this discrete distribution of variances the **variance map**. The variance map is later used to extract interesting regions in the data (as described in Section 3.2.2), and also surfs as the basis for our visual exploration.

3.2.2 Automatic High Variance Localization

To guide the users during the exploration, we extract h surface regions showing the highest error variances from the variance map. These surface regions are assumed to be the most interesting areas for comparison due to the high disagreement between the reconstruction methods. We extract these regions automatically in the data and present them later to the users. The detection of high-variance regions in the data only has to be done once in a pre-processing step, directly after the variance map computation.

The variance map computed in the previous step serves as a basis for finding *local maxima* in the distribution of per-vertex variances on the surface. These maxima are found by employing a weighted Mean-Shift [CM02] algorithm in $\mathbb{R}^3$ on the set of mesh vertices. In our case, the weights for the weighted Mean-Shift algorithm are defined by the per-vertex variances σ_v^2.

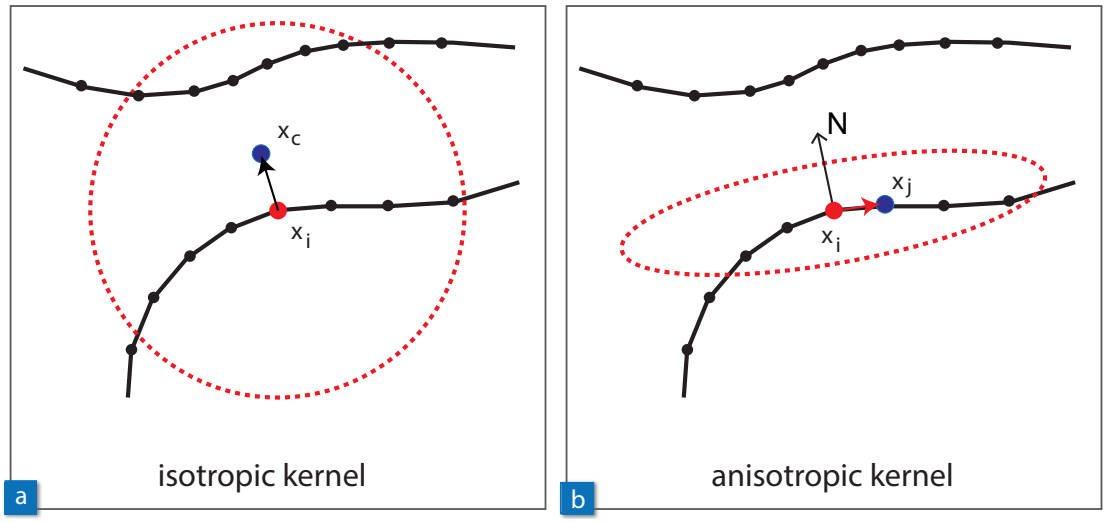

Figure 3.3: Surface-aware Mean-Shift. Instead of using an isotropic kernel (a), we employ a surface-aware Mean-Shift with an anisotropic kernel (b) around the current mean x_i. This prevents the means from moving away from the model surface (x_c). Instead, by considering the normal vector N, the means stay close to the intrinsic surface (x_j).

A set of initially random samples x_i, $i = 1 \ldots k$ of mesh vertices are iteratively shifted toward modes in the variance distribution using a smooth kernel $\theta(r) = e^{-4(r/s)^2}$ of finite support s. One iteration step is given as

$$x_i' = \frac{\sum_j \theta(\delta_{ij})\sigma_j^2 \, p_j}{\sum_j \theta(\delta_{ij})\sigma_j^2} \tag{3.2}$$

where σ_j^2 gives the variance at vertex j, while p_j is its location in $\mathbb{R}^3$ and δ_{ij} denotes its distance to the sample x_i.

However, an isotropic kernel might let samples move away from the intrinsic surface described by the local neighborhood of vertices (Figure 3.3a). Thus, we need to constrain the sample movements to be close to the local surface around x_i (Figure 3.3b). We employ a *surface-aware* distance metric δ_{ij}, which incorporates the surface normal n_i into the weighting kernel as given by

$$\delta_{ij} = \left\| p_j - x_i \right\| + \left| \left\langle n_i, p_j - x_i \right\rangle \right|. \tag{3.3}$$

After the samples converged to different high-variance modes on the surface, spatially similar points are discarded, and the remaining ones are sorted by their amplitude. This gives a list of **hot-spots** that we use in the consecutive visual analysis procedure. Besides the positions p_i of the hot-spots, we are also interested in their extent, which is given by the weighted sample standard deviation σ of variance values at every hot-spot.

3.2.3 Parallel Coordinates Plot

The list of hot-spots created in the previous Section can be used to span a multi-dimensional feature space. The space is defined by the number of hot-spots h and the error values e_n for all n input meshes at the hot-spot positions. We propose to use the high-dimensional visualization technique of *parallel coordinates* [ID90] to analyze this multi-dimensional feature space.

Every hot-spot defines one axis in the parallel coordinates plot, and the dimensions of the axes are given by the global minimum and maximum error values. The input meshes are represented as polylines in the plot. A schematic representation of the parallel coordinates plot can be found in Figure 3.4. The axes in the plot are initially sorted according to the hot-spots' weighted sample standard deviation of variance values. This assures that the hot-spots at positions where the input meshes produced varying results are listed first. The sorting of the axes can be interactively changed by the user. The parallel coordinates plot is initialized with the pre-computed hot-spots. Users can also manually insert new hot-spots into the plot which are especially interesting for their analysis.

The parallel coordinates plot gives an abstract representation of the information encoded by the hot-spots. The plot enables users to track the error rate of algorithms in different local regions in the data. This reveals whether certain algorithms produce desired results in all regions of interest, or just in a subset of them.

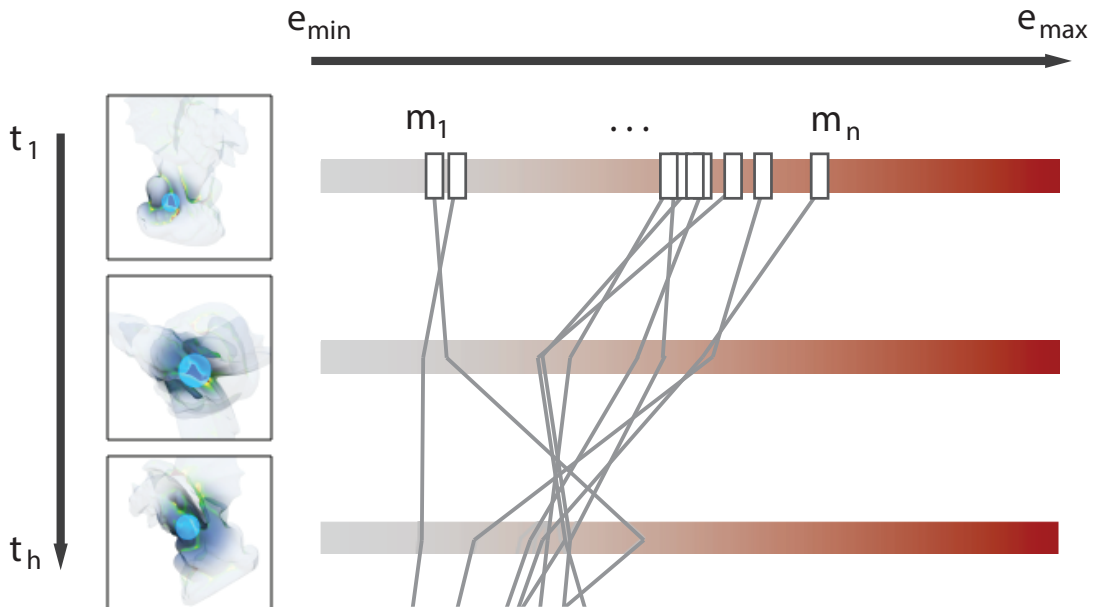

Figure 3.4: Parallel coordinates plot. We can use the hot-spots to create a multi-dimensional feature space, defined by the hot-spots $t_1 \ldots t_h$ and the error values e_n at those positions for all n input meshes. The hot-spots are represented as thumbnail images, and the input meshes $m_1 \ldots m_n$ are defined by polylines.

3.2.4 Visual Analysis

YMCA provides interactive tools to explore the results of the mesh comparison analysis. The interaction concepts follow the principle of *overview first*, *details on demand*. The main elements of the user interface are illustrated in Figure 3.5.

To provide an **overview** of the differences in the data, we propose a rendering of the reference mesh (Figure 3.5a). A *heat map* is projected onto the mesh according to the current variance map. The default heat-map colors range from blue (low variance) to red (high variance). The decision to choose a color scale ranging from blue to red was the result of some discussion with our collaborators, because such color maps are widely used in practice, and they were already familiar with such a heat-map color scale. We are aware that the color scale may not be ideal (e.g., for colorblind people). If necessary, the color scale can therefore be adjusted by the users by selecting different colors for the minimum and maximum variance values. The reference mesh rendering allows users to inspect differences in the data without having to check all individual meshes, because the relevant information is aggregated in one view. An example for an overview image can be seen in Figure 3.6.

In the user interface, the *hot-spots* are arranged in a parallel coordinates plot. The **parallel coordinates plot**, where the hot-spots are embedded in, represents the input meshes as polylines, indicating their local error value at the hot-spot positions (Figure 3.5b). To interact with the data, the users can change the ordering of the axes and also eliminate individual hot-spots by mouse interaction. Individual polylines can be activated and selected. It is also possible to create new hot-spots during the analysis. To give users an idea about the position and size of the hot-spots, the corresponding surface regions are represented by *thumbnail images* of the mesh. The thumbnails are displayed at the corresponding parallel coordinates axes. The thumbnail images are created when a hot-spot is generated in the system. The reference mesh is used to produce the images, and the viewports are given by the corresponding hot-spots' locations. The users can interact with the thumbnails by mouse. When clicking on one of them, the overview is automatically rotated to the location of the hot-spot. The optimal viewing angle is calculated by aligning the viewing direction with the normal vector of the hot-spot and centering it in the viewport. To emphasize the hot-spot locations, we use a focus+context approach in the rendering of the mesh. We employ opaque rendering only in the hot-spot area, while the rest of the mesh is depicted with high translucency (Figure 3.5c). The users can employ the parallel coordinates plot and the hot-spots to quickly depict input meshes containing undesired results and eliminate them from further analysis. It is also possible to compare input meshes based on local properties.

Besides giving an overview of the data, YMCA also provides means to further inspect local variations. We propose a **magic lens** tool (Figure 3.5d) which can be used to select a certain region of interest on the surface of the mesh. The magic lens is circular, because drawing a circle is a very intuitive way of selecting objects. A colored, transparent circle drawn over the mesh indicates the current location of the lens. The size can be dynamically adjusted by mouse interaction. Since the circle is transparent, the selected part of the reference mesh remains visible in the 3D view. The magic lens is linked to the detail view (as described below), so that the properties of the input meshes can be explored at the current position of the lens.

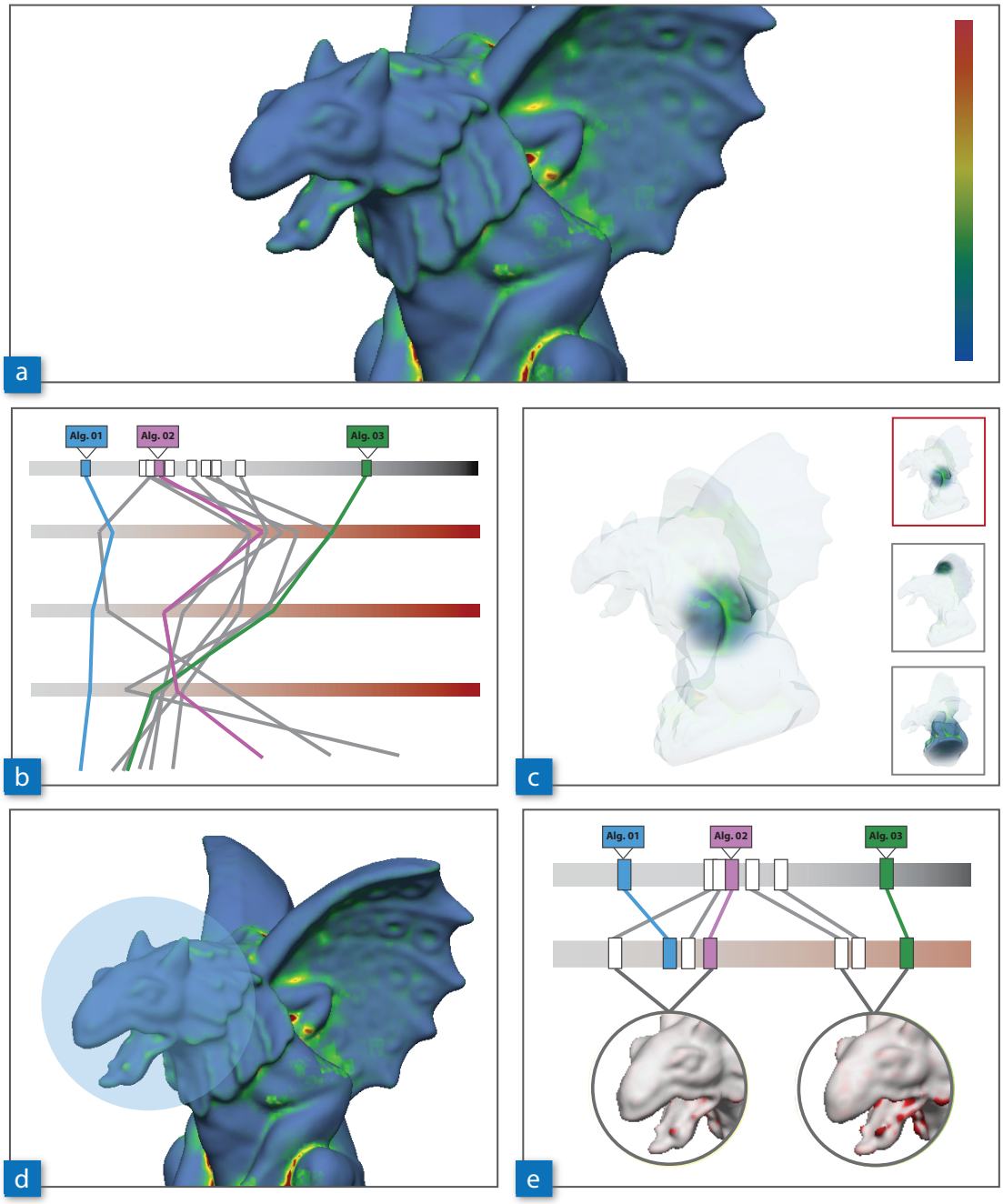

Figure 3.5: User interface elements. YMCA conveys an overview of the available data (a) by rendering the previously computed variance map. Hot-spots are computed, which point to regions of high variance in the data. They can be arranged in a parallel coordinates plot to compare the input meshes based on local regions (b). Rendering of the hot-spots points users to interesting features (c). The local regions can be explored by employing a magic-lens (d), which updates the corresponding detail view (e).

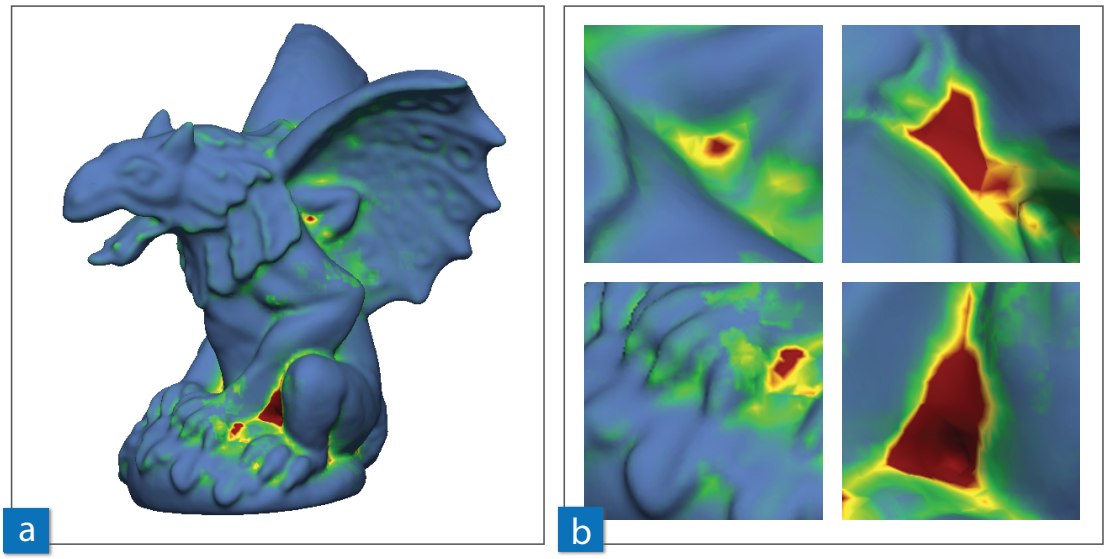

Figure 3.6: Overview image. In this figure the reference mesh with a projected heat map according to the variances in the data is shown (a), as well as some mesh regions in more detail (b).

The current location and size of the magic lens tool is used to present more detailed information about the local behavior of the mesh reconstruction algorithms in a **detail view** (Figure 3.5e). A more detailed example of how a detail view can look like is given in Figure 3.7. To provide quantitative information, a *ranking* of the mesh algorithms is provided. The reconstruction algorithms are sorted according to their corresponding accumulated error at the current lens position. For every algorithm a rectangle is placed at the corresponding position along an error scale (Figure 3.7a). The scale ranges from the global minimum to the global maximum error. In addition, the ranking of the algorithms at the current lens position is shown below (Figure 3.7b). This reveals whether an algorithm, which has a low/high global ranking, produces different results at the current local position. The users can activate rectangles by mouse interaction to reveal the name of the algorithm at this position. In addition to the ranking, further visual information is needed for the analysis of the meshes, for which we added a *data summarization* view (Figure 3.7c). Here the meshes are *classified* according to their accumulated error at the current lens position (see Section 3.2.5 for further information). This summary gives an overview of the variance and possible problems at the current lens position. According to feedback from domain experts, besides having an overview, it still is necessary to have access to the individual meshes. Therefore, we allow users to see *close-up views* of the reconstructed meshes. If more than one mesh is selected, we place the close-up views in a juxtaposition (Figure 3.7d). The meshes inside the close-up views are color-coded according to the accumulated error at the current lens position. Here we use a different heat map than the one projected onto the reference mesh to make a clear distinction between the variance and the local error values. All interface items are updated every time the magic lens is moved or resized.

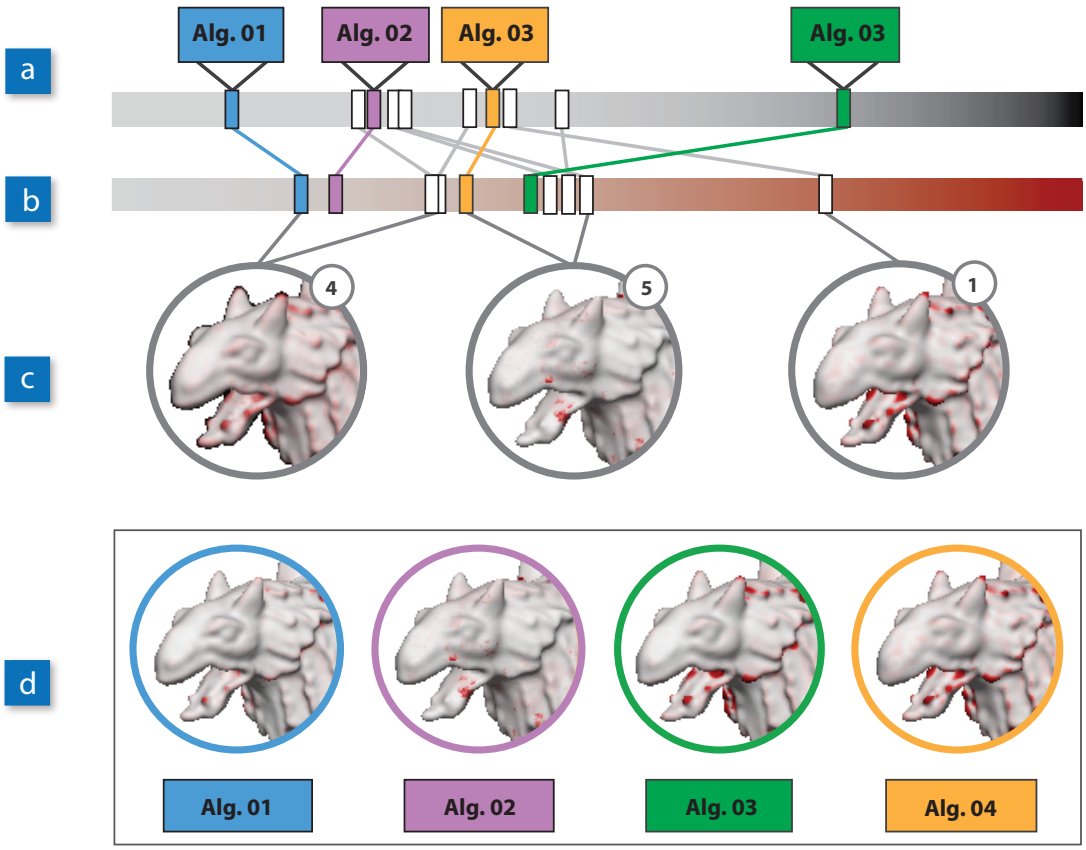

Figure 3.7: Detail view. When using the magic lens tool, the users can hover over the reference mesh and inspect parts of it in more detail. The global error value of the input meshes (a) and the local error at the lens position (b) are displayed at the top. The meshes are classified according to their local error value at the current position of the lens (c). It is also possible to view individual mesh renderings (d).

We provide some **additional controls** which can be used to adapt the system's interface elements according to individual preferences. As mentioned above, the color scales of the overview image and the close-up views of the reconstructed meshes can be customized by the user. In addition, the upper and lower bound of the color heat map in the overview image can also be adjusted, which allows users to concentrate on different variance ranges. The render mode can be changed from hot-spot rendering to heat-map rendering at any time. In the detail view, the users can decide whether they want to see the global ranking as well. The detail view can further be adjusted to concentrate on the data summarization, on the individual meshes, or both. It is possible to replace the reference mesh by some other input mesh. Then all differences and variances have to be re-computed, and the data has to be re-loaded. This option can be employed by the users to compare one mesh with the other meshes in the dataset (e.g., to evaluate a new mesh reconstruction technique).

3.2.5 Data Summarization in the Detail View

To deal with scalability and to provide a condensed overview of the data, we decided to integrate a data summarization into the detail view. If hovering with the magic lens over the mesh, the users should get details about the underlying data, as well as a combined summary. The purpose of the summary is to quickly inform the users about the variability of the data at the current lens position and to indicate whether further inspection would be necessary (e.g., if meshes exhibit varying results).

We propose to **classify** the reconstructions according to their accumulated error at the current lens position. The variance at the current lens position indicates the number of classes which are needed. After detailed discussions with our collaborators we came to the conclusion that dividing the data into three classes (best/middle/worst) is a very intuitive way of presenting a summary of the data. Therefore, a maximum of three classes is allowed. We use two fixed thresholds that define the final number of classes. An average image is used as class representative to display the data. Figure 3.8 gives an example of how the data summarization could look like. If the variance at the current lens position is low, only one class is created containing all meshes (Figure 3.8a).

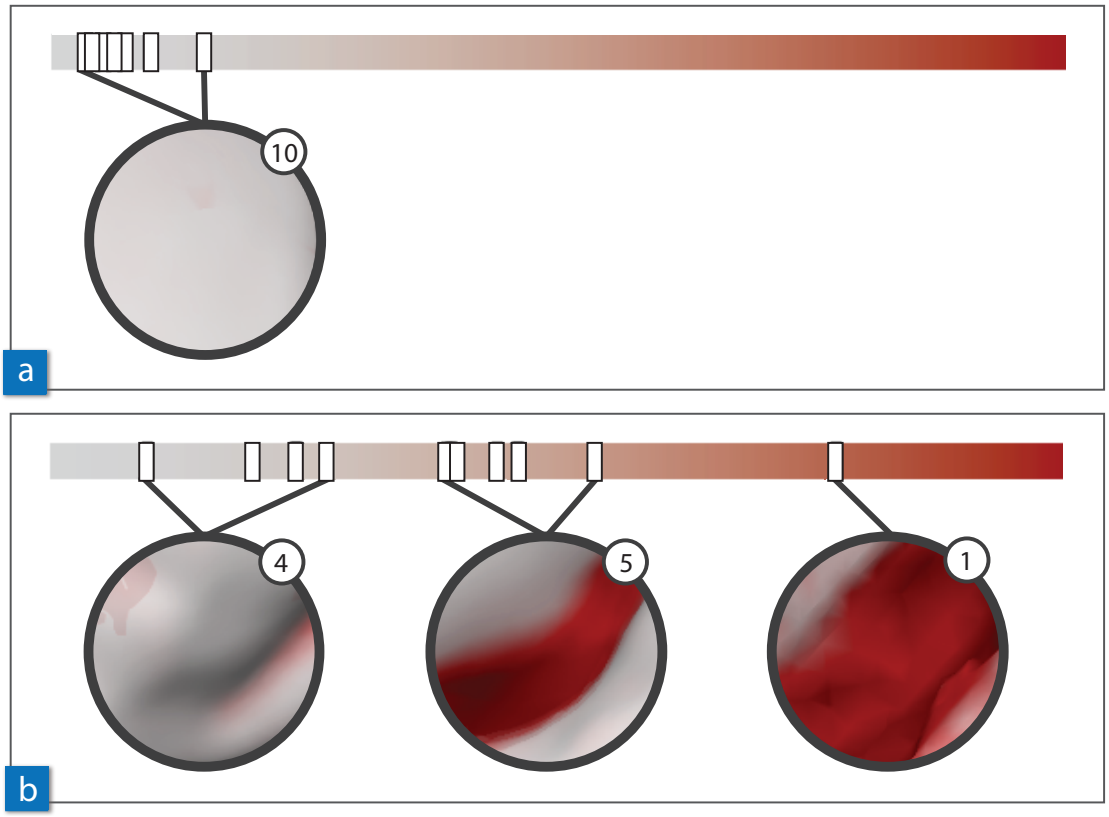

Figure 3.8: Data summarization. This figure shows two examples for data summarization in the detail view when hovering over areas of low (a) and high (b) variance.

This shows that all reconstruction algorithms produced the same result at this position. The higher the variance, the more classes are created (Figure 3.8b). Such positions on the mesh, where the reconstruction algorithms produced very different results, might need further visual inspection by the users.

The proposed data summarization provides a good overview of the data, where the users can quickly decide about a further local inspection of the data. In addition to this level of abstraction, the average images representing the classes still reveal the underlying information.

3.2.6 Hot-Spot Rendering

The hot-spots in YMCA can be dynamically activated in the user interface by selecting a hot-spot thumbnail in the parallel coordinates view (Section 3.2.3). This automatically rotates the overview to an optimal viewing angle to uncover the region of interest on the reference mesh. However, in many cases the hot-spots are located in concavities of the surface, which are often occluded by other parts of the mesh. Thus, a clear view onto the hot-spot may be prevented, and the users might lose the focus if they rotate the model.

We therefore use a visualization technique that removes any occlusions of the interesting surface region by **increasing the transparency** with the distance to the hot-spot. Given a pre-computed hot-spot position p and its extent σ, we employ a smooth transparency kernel $K(x) = e^{-4\|x-p\|^4/\sigma^4}$ to put the hot-spot into focus (full opacity) while removing occluding surface parts and at the same time providing background context (high transparency). This is done by ray casting, using two render passes: First, in an accumulation pass, the whole mesh is drawn into a texture using accumulative blending. Every fragment with corresponding surface position x is weighted by the kernel $K(x)$. This way, the resulting texture stores for each pixel the weighted sum of surface colors along the respective ray, as well as the sum of all weights. Then, in the normalization pass, the accumulated values in the texture are normalized by the sum of their weights and drawn onto the screen (Figure 3.9).

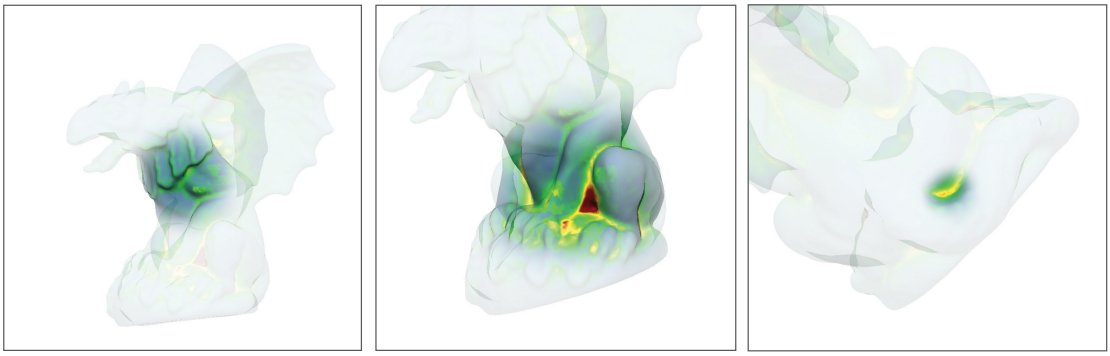

Figure 3.9: Hot-spot exploration. This figure shows a model rendered in hot-spot mode (Section 3.2.6) with three different hot-spot examples (different position and extend).

3.3 Implementation

The pre-processing step, consisting of comparing the meshes and calculating the variances, was implemented in C++. We used *MeshDev* [RFT04] to calculate the differences. The cost of the pre-processing step, consisting of the calculation of the variance map and the location of the hot-spots, depends on the number of meshes in the input data set. No user input is required during the pre-processing step. The interaction itself, which works in real-time, has been implemented in C++ and OpenGL/GLSL. The application was tested on a machine with an Intel i7 CPU, 12 GB of RAM and an NVIDIA GeForce GTX 580 graphics card. A comparison of the computation times and memory requirements during the analysis can be found in Table 3.1. A more detailed description of the datasets can be found in Section 3.4.

Table 3.1: Runtime and memory requirements. In this table the dataset sizes, the runtime for the pre-processing and the memory requirements are shown. The first column gives the name of the dataset. The second column shows the number of meshes the corresponding dataset comprises of. The next column depicts the runtime for the pre-processing, which consists of the calculation of the variance maps and the localization of the hot-spots. The last column shows the amount of memory used on the graphics card.

Dataset	**Meshes**	**Pre-Processing**	**Memory**
Gargoyle	10	12.7 s	82.6 MB
Dancing-Children	100	86.9 s	797.5 MB
Daratech	15	17.2 s	131.4 MB

3.4 Results

Reconstruction algorithms perform with varying accuracies on different parts of a surface. YMCA allows users to analyze these differences. The parallel coordinates plot in conjunction with the hot-spot thumbnails enables users to understand the relative performance of different algorithms on a particular part of a surface, and at the same time allows for the visual inspection of the reconstructed surface and its quality.

We used data from the field of point-cloud reconstruction to test our approach. The data was produced by different algorithms, for example Poisson Surface Reconstruction (*Poisson*), Algebraic Point Set Surfaces (*APSS*), and Multi-level Partition of Unity Implicits (*MPU*). The reader is referred to the survey by Berger et al. [BLN+13] for a more detailed description of the reconstruction algorithms. We applied our approach to three different datasets. The first dataset, called *Gargoyle*, comprises ten mesh reconstructions from a virtual point cloud scan of a carved

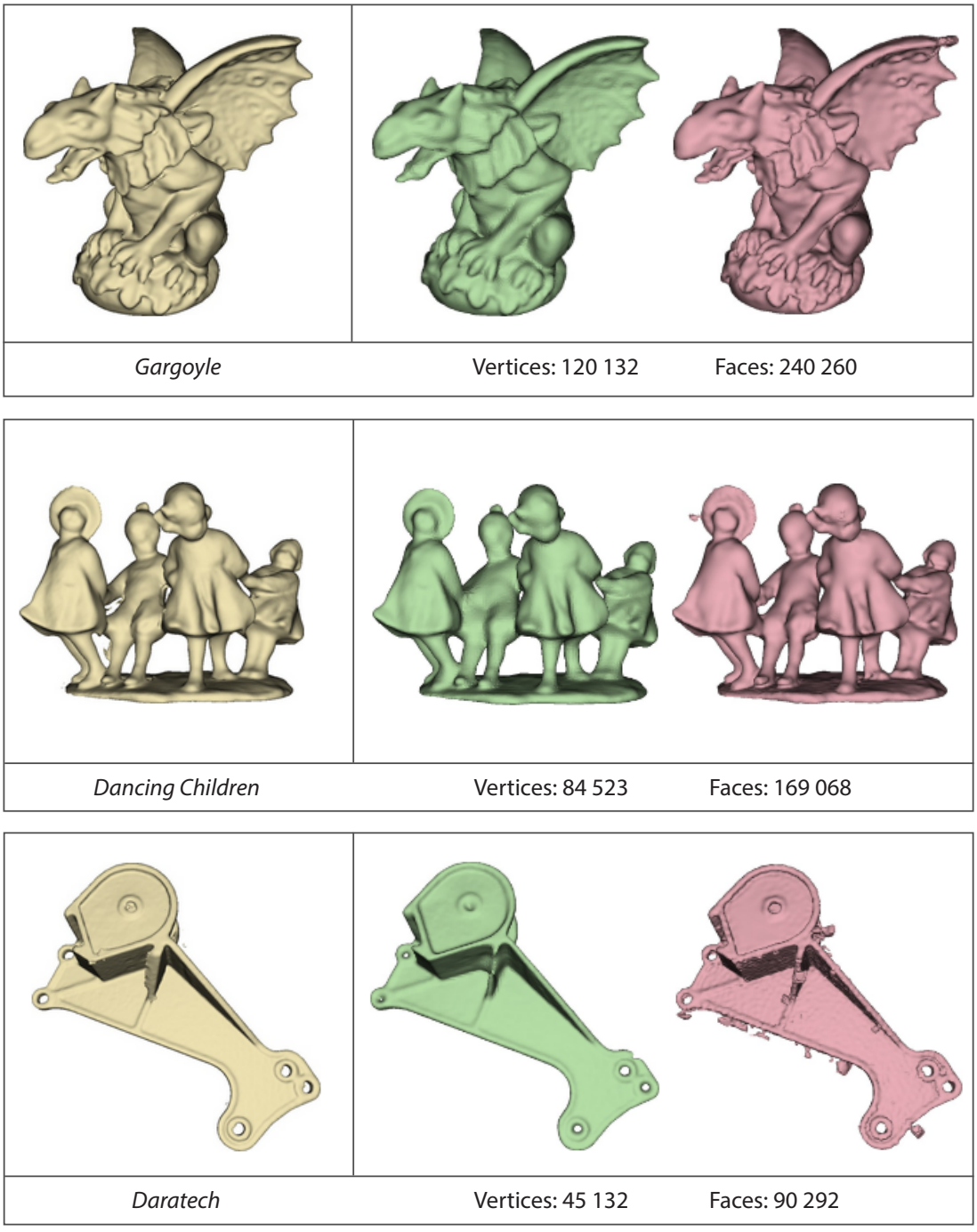

Figure 3.10: Datasets used to evaluate YMCA. All datasets consist of mesh reconstructions from point cloud scans. The reference mesh of the datasets and two selected reconstructions each are shown. The *Gargoyle* datasets shows the scan of an ancient statue, the *Dancing-Children* a scan of a small figurine, and the *Daratech* datasets represents the scan of an industrial workpiece [BLN+13].

stone figure. The second dataset, called *Dancing-Children*, consists of 100 mesh reconstructions from a virtual point cloud scan of an figurine. The third dataset, called *Daratech*, comprises 15 mesh reconstructions from a scan of an industrial workpiece. Figure 3.10 shows renderings of the reference meshes, examples for reconstructions, and further information about the mesh dimensions of the three datasets.

In all datasets, parts of the variance map exhibit rectangular-shaped artifacts. We used the overview image and the magic lens to further inspect those areas. With our tools we could find out that these artifacts are always produced by the *Poisson* reconstruction algorithm (Figure 3.11). Apparently, this artifact is caused by the limited resolution of the octree employed by the *Poisson* algorithm for reconstruction. With the visual analysis tools of YMCA, this artifact could quickly be related to the *Poisson* algorithm and explored visually. It is clearly visible in which part of the models the octree resolution has to be adjusted to guarantee a smooth reconstruction.

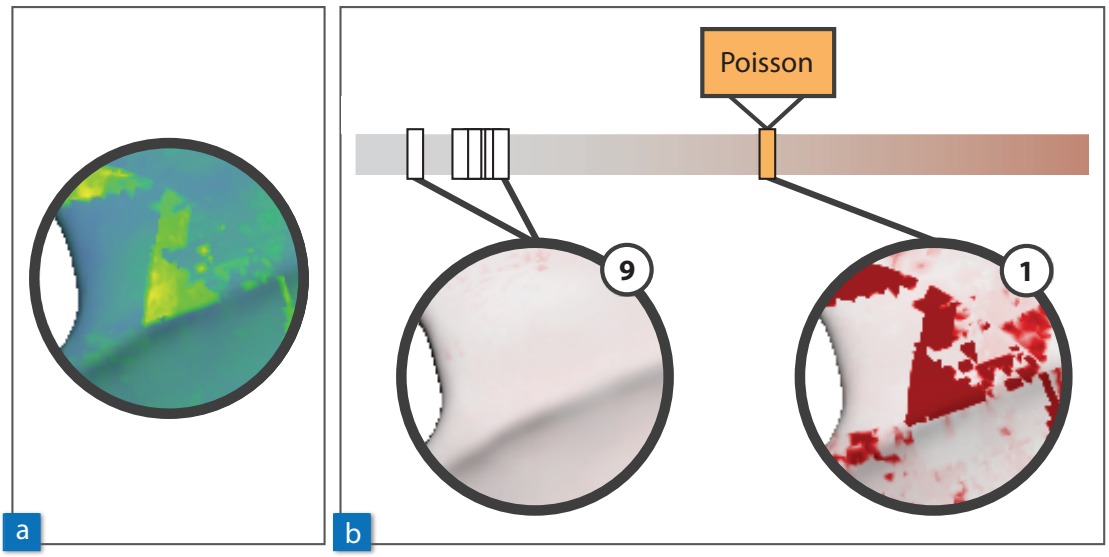

Figure 3.11: Artifact analysis. In all datasets rectangular-shaped artifacts could be identified in the overview image (a). With the magic lens tool it is possible to find out that these artifacts are caused by the *Poisson* reconstruction algorithm (b).

When analyzing the *Daratech* dataset, we could identify artifacts on the surface which were produced by the *Wavelet* and the *Scat* algorithms. This is shown in Figure 3.12a and 3.12b (wrong mesh vertices are highlighted in red). The artifacts are created due to the presence of noise in the data. YMCA clearly classifies these algorithm as individual local outliers, with relatively high reconstruction error. At a different part of the model, the *RBF* algorithm stands out by wrongly closing a hole, where all remaining algorithms perform correctly (Figure 3.12c). By giving this integrated overview of the algorithms' relative performance in different surfaces areas (statistically and visually), the hot-spot localization and the parallel coordinates plot allow the users to quickly classify algorithms and judge their eligibility. As mentioned before, a manual comparison of all meshes would be far more tedious and can lead to particular artifacts being easily missed.

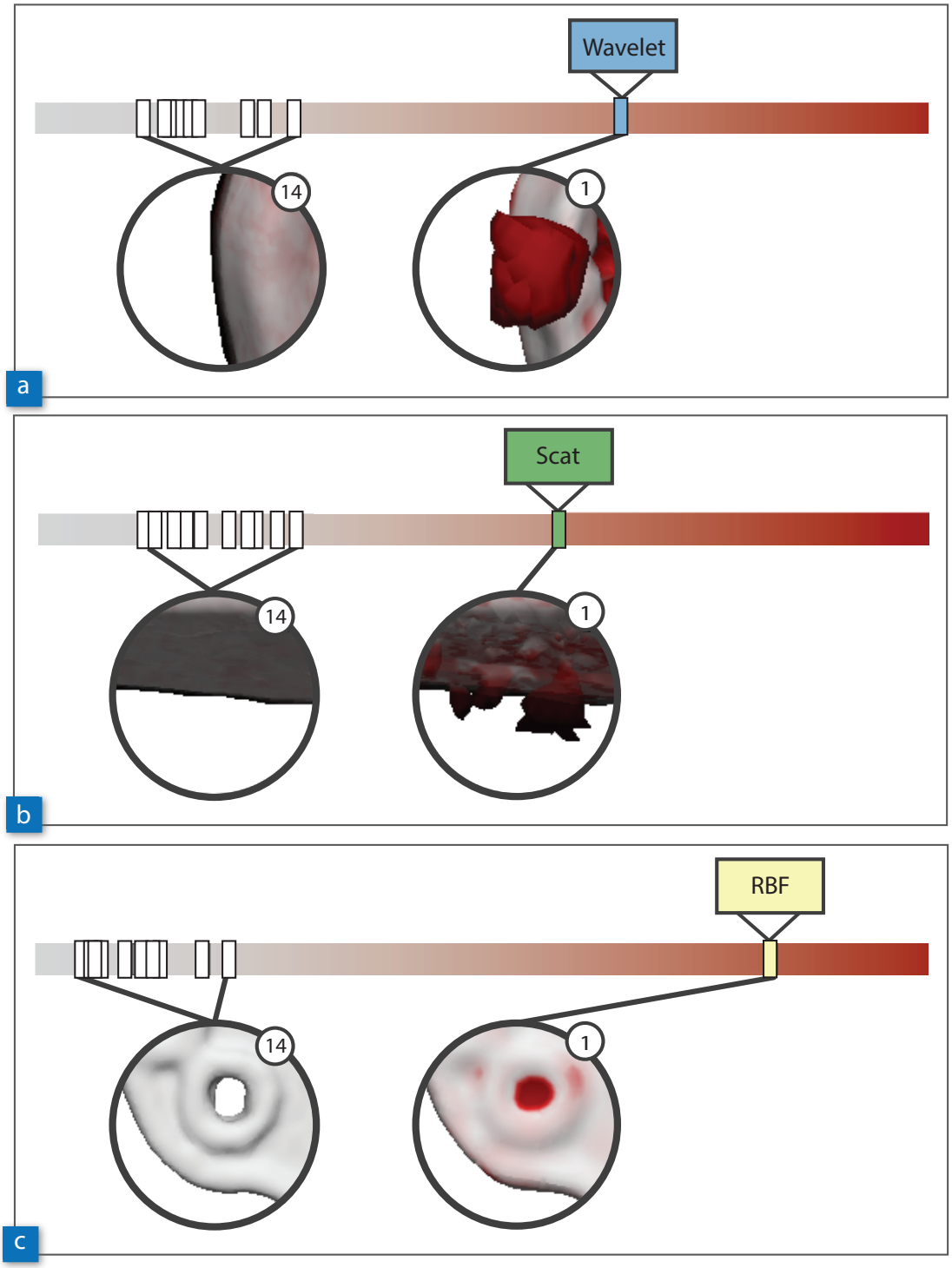

Figure 3.12: Outlier detection. The summarization in the detail view of YMCA allows users to quickly detect outliers, which might be caused by noise in the data (a,b) or by a certain algorithm behavior like over-smoothing (b).

A very interesting and helpful feature of YMCA is that with the data summarization used in the detail view, differences at the reconstructed mesh boundaries can be explored. Blending the lens view of meshes of the same class allows for a direct comparison and visualization of the geometric variance of their boundaries. Different boundaries in the reconstruction result due to different approaches to fit a surface to the point cloud data. In Figure 3.13, two examples of different boundaries can be seen. Many parts of the boundaries of the *Gargoyle* model have been reconstructed in a similar way by all algorithms. In this case no differences at the boundaries are visible (Figure 3.13a). In other parts of the model, the boundaries of the reconstructed meshes differ more strongly. In this case the differences in the boundaries become visible as a color band in the images (Figure 3.13b). The *Gargoyle* model contains a lot of curved surfaces, and here the boundaries of the reconstruction algorithms differ the most. With YMCA it is now possible to explore these effects in detail. The regions where boundaries differ are clearly visible when inspecting the mesh with the magic lens. This helps the users to judge which reconstruction would better represent the data.

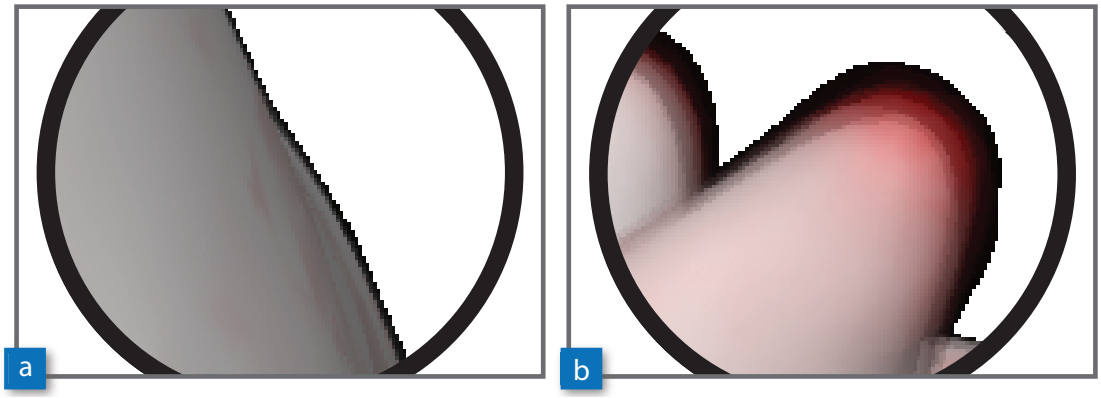

Figure 3.13: Mesh boundaries. The summary view in the detail view enables users to identify regions where the reconstruction algorithms produce almost the same (a) or different (b) mesh boundaries, indicated by a color band.

One major focus of our work for mesh comparison was scalability pertaining the number of ensemble members, so that the application can be used for large mesh ensembles. The parallel coordinates plot of YMCA proved to be very helpful in the analysis of large datasets. If such large datasets shall be inspected, users typically would like to quickly narrow down the search space. Then the users can concentrate on the reconstructions of interest in more detail and eliminate others from further analysis. Depending on the task, users might want to quickly spot reconstruction algorithms that perform either generally very good, or very bad. The parallel coordinates plot can be employed to identify these cases.

Users can activate individual or groups of input meshes in the parallel coordinates plot. An example is given in Figure 3.14. It shows the parallel coordinates plot as it look like when inspecting the *Dancing-Children* dataset, which consisted of 100 ensemble members. In the

parallel coordinates plot, it was possible to identify one algorithm (*Fourier-3*) with a low error rate in the available hot-spot locations. The algorithm also exhibits a rather low global error rate. One reconstruction algorithm could be spotted with a high global error rate, and which also peforms rather bad in the available hot-spot regions (*SPSS-7*). Depending on the task, the users might want to eleminate such algorithms with a good/bad performance from further analysis. As a third example, we could identify one algorithm that exhibits a varying error rate in the available hot-spots (*Scattered-2*). In the case the users would have to visually inspect the mesh properties of the reconstruction algorithm at the hot-spot positions, to be able to judge whether the algorithm meets the reconstruction requirements.

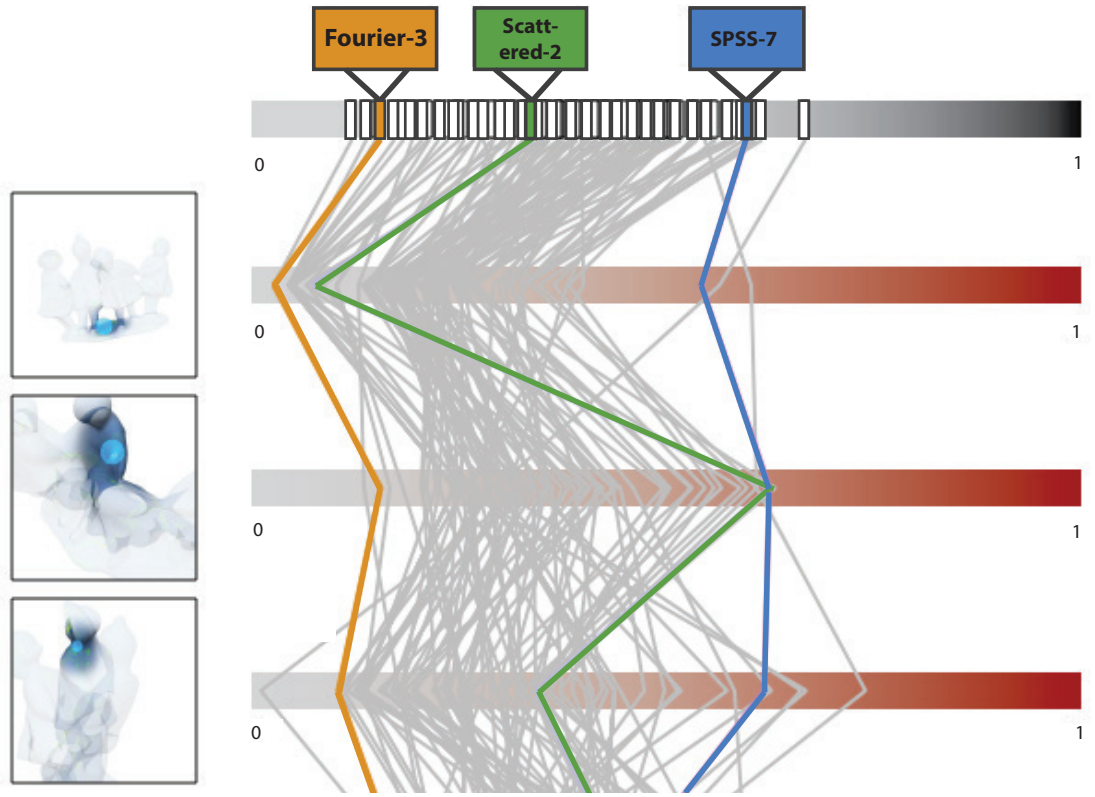

Figure 3.14: Analyzing large datasets. The parallel coordinates plot is a very helpful tool when analyzing large datasets. One algorithm with a low error rate (*Fourier-3*), one with a high error rate (*SPSS-7*) and one with a varying error rate (*Scattered-2*) in the hot-spot regions could be identified very quickly.

3.5 Evaluation

To evaluate our approach, we have collected qualitative feedback from users experienced in working with meshes. From this feedback we wanted to find out how useful the proposed visual analysis is for the experts, and we also discussed possible applications for future work.

Every **feedback session** lasted between 30 and 45 minutes. First the motivation, the YMCA application, and all interaction techniques were explained to every participant. Then the participants were provided with a training dataset where they could test the interaction possibilities and get familiar with the application. Afterwards they were presented a new dataset, and were asked to name one or more reconstruction algorithms that produce appropriate results for the given point cloud. They were also asked to explain their decision at the end. Participants had ten minutes time to finish the task. We used the same training and evaluation datasets for all participants. At the end of the evaluation session, we asked them to fill in a questionnaire with four questions.

We asked seven **participants** (six men, one woman) to participate in our feedback study. Three of the participants have been working in the field of point-cloud reconstruction for years, and therefore have a lot of experience. Two other participants are experienced computer scientists in the rendering field, where they are working with mesh operations like filtering or simplification. Two participants are students from the field of computer graphics. All participants agreed that analyzing point cloud reconstructions is an important task, and that existing methods do not provide sufficient assistance for this. Six out of seven participants had no problem to solve the task of finding an appropriate reconstruction algorithm for the given point cloud. One participant ran out of time while solving the task. We compared the results with reconstructions that were previously selected by domain experts. It turned out that participants selected the most suitable reconstruction algorithms in all cases.

With the first three questions in the questionnaire we wanted to find out more about the **practicability** of the system:

1. *Question 1*: Does the system help to spot point cloud regions which are problematic for reconstruction?
2. *Question 2*: Does the system help to decide which reconstruction algorithm should be used?
3. *Question 3*: Does the system help to better understand the strengths and weaknesses of certain reconstruction algorithms?

The answers to these three questions can be seen in Figure 3.15a. The participants agreed that YMCA helps to spot the most problematic regions in the reconstruction from a point cloud (*Question 1*). They also largely agreed that the system can help to identify the most appropriate reconstruction algorithms for a given point cloud (*Question 2*). However, they were discordant about whether YMCA helps to better understand how the reconstruction algorithms work (*Question 3*). Some participants stated that algorithms can be judged in a visual way, but for a detailed analysis additional information about the point cloud (i.e., noise level) would be necessary. Also

some more information about the algorithm would then be needed. For future work, it would be interesting to analyze the algorithms' pipelines in more detail, and to also take into account the influence of different parameter settings.

The fourth question in the questionnaire concerned which elements of the user interface the participants found **helpful** during the analysis. We asked which of the following elements they used the most:

- Variance map (i.e., overview image)
- Parallel coordinates (i.e., visualization of reconstruction results in the parallel coordinates plot)
- Hot-spot localization (i.e., the possibility to click on hot-spot thumbnails in the parallel coordinates plot and the hot-spot rendering mode)
- Detail view and data summarization (i.e., detail view with close-up views and ranking, and data summarization)

The answers to this question can be found in Figure 3.15b. The overview image showing the variance map was rated to be the most helpful interface element for the users. This is not surprising, since at the one hand this is the central interaction element of the system, and on the other hand most users are already familiar with interpreting color heat maps on 3D models. The parallel coordinates plot was very helpful for the participants to compare the overall and local performance of individual algorithms. They used this interface element especially to eliminate reconstructions from further analysis. Although all users were positive about the fact that a list of hot-spots is already prepared when starting the analysis, some of them did not like the hot-spot rendering technique. They stated that it is confusing at the beginning and needs more experience to be interpreted in the right way. The participants also used the detail view to judge the local behavior of the algorithms. Only one participant stated that the data summarization is sometimes hard to interpret and would need a longer training period.

We also asked the participants about suggestions for **future work**. During these discussions it turned out that the system inspired the users quite a lot, and they had many suggestions for additional features and applications. For the overview image, one participant stated that it would be helpful to see the reference mesh rendered in colors according to the algorithms which perform best at certain points. This could be a valuable hint in the analysis. Having a very colorful model means that the inout algorithms differ a lot, whereas having large uniformly colored parts means that one algorithm performs better than all others in those areas. For the data summarization in the detail view, the participants suggested that sometimes it would be useful to be able to manually adjust the class borders, or even do a classification by themselves (by manually selecting algorithms). The participants experienced with point cloud reconstruction also stated that they liked the data summarization in the detail view, because it quickly provides an overview of the data at the current local position. They also pointed out that summarization alone is not helpful for them. To be able to judge which algorithm performs better than the others, they still need to be able to access the individual input meshes. Therefore, they also liked the possibility to depict all close-up views of all meshes from one class in a small multiples display.

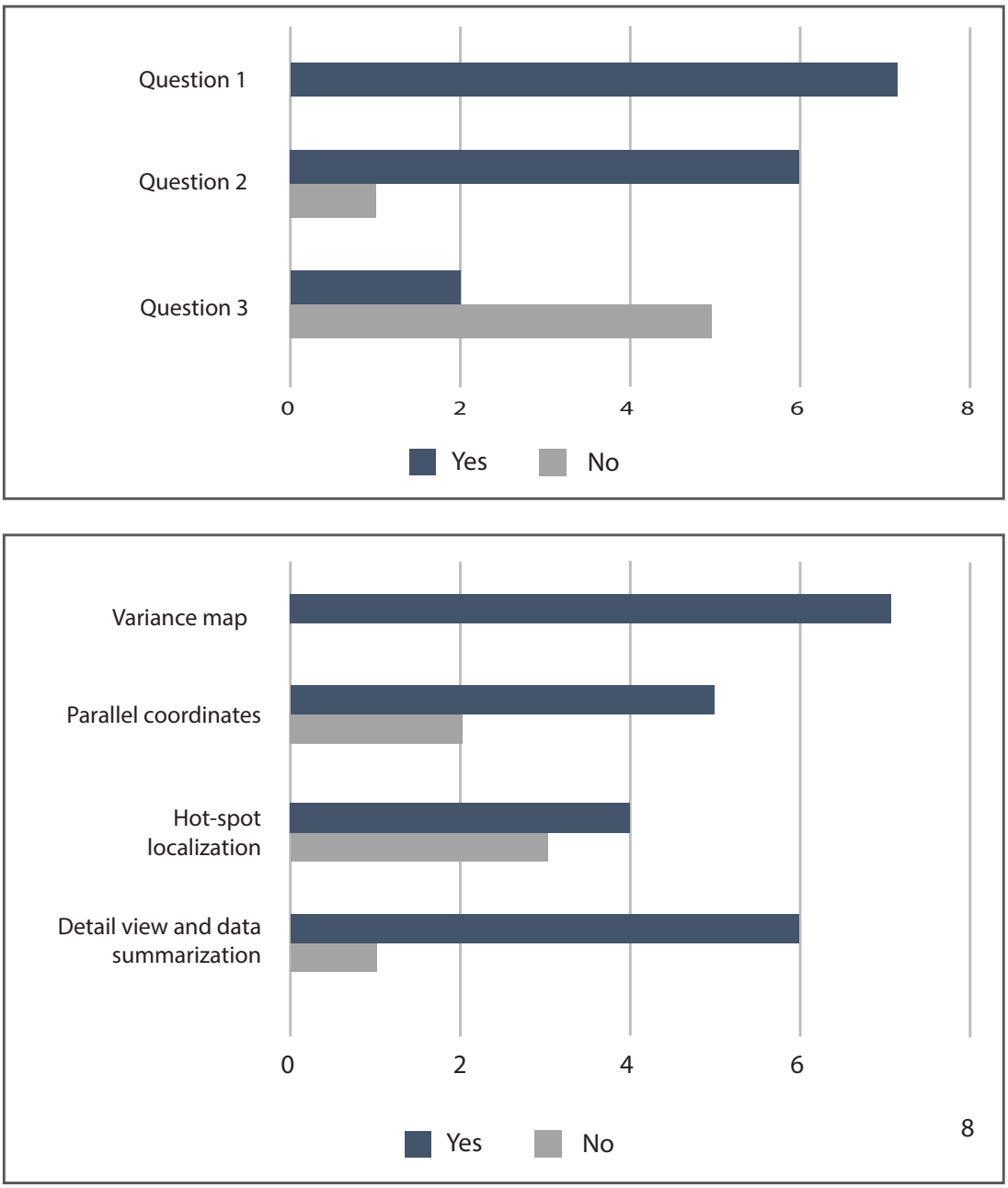

Figure 3.15: Evaluating YMCA. First we asked three questions to evaluate the practicability of our system. YMCA proved to be very helpful for evaluating point clouds and finding the most suitable reconstruction (*upper chart*). In the second part of the evaluation we wanted to find out which user interface elements the users found the most helpful ones (*lower chart*). All participants liked the variance map and rated it to be very useful. Also the parallel coordinates plot and the detail view were used by most of the participants. However, the hot-spot rendering was sometimes confusing and therefore not rated to be very useful in all cases.

3.6 Summary

YMCA is a visual analysis application for the comparative visualization of multiple 3D meshes. Not many suitable tool-sets existed so far that allowed for the efficient comparison of multiple meshes. YMCA provided interactive tools to present an overview of the differences in the data, and to explore local areas of interest in more detail. Our visualization approach combines explicit encoding, juxtaposition, and parallel coordinates. It further addresses the scalability problem of previous mesh comparison tools. We applied our approach to meshes coming from different mesh reconstruction algorithms that were applied to point clouds. With our method, differences between several resulting meshes can be quickly identified, and it is also helpful to explore individual characteristics of the different mesh reconstruction algorithms. With YMCA it is now possible to quickly and visually analyze mesh reconstruction results, and depict the appropriate solution for the given point cloud data. Our system nonetheless has some limitations, which are discussed in this section, because they point to interesting directions for future work.

For the data tested in the scope of this thesis, **reference meshes** were given which could be used to calculate the differences in the meshes. This, however, this is not the case when analyzing point cloud data (e.g., if scanning real-world objects). In this case we propose to create an average mesh out of the input meshes, and to compare the meshes with this average. This provides an initial overview of the differences in the data. If users are not satisfied with comparing the meshes with the average, they can exchange the reference and use some other input mesh instead, e.g., the mesh that delivers the best reconstruction of certain parts of the input data. YMCA already provides controls for exchanging the reference mesh.

We used an attribute deviation metric [RFT04] to calculate the mesh differences. YMCA is primarily about the visual depiction of mesh differences, which means that the calculation of the differences is decoupled from the visual representation. The **metric** can be exchanged with other vertex-based mesh difference calculation. We tested this by using geometric deviation [RFT04], which can be seen in Figure 3.16.

The variance map of YMCA gives an overview of the regions in the point cloud where the reconstructions produce different results. However, for some applications it might be interesting to see the **global error** instead. This would better reveal regions where all reconstructions fail. To test this, we used the mean squared error to aggregate the different errors into one value per vertex. YMCA already provides means for exchanging the metric, which can be also applied to using the error values for visualization instead of the variances. At present this means that the system has to be initialized with either the one or the other settings. The users can only explore the results of one metric at a time. This is something that we want to change in the future. We would also like to work on possibilities where the metric can be changed during analysis, while the settings (like selections in the parallel coordinates plot) are still preserved for all metrics. This way it will be possible to select a list of hot-spots (calculated by different metrics) and use them for further analysis.

As pointed out during the evaluation, YMCA provides limited support in understanding the **strengths and weaknesses** of individual reconstruction algorithms. Our approach enables the users to visually compare the results and therefore judge them, but domain experts stated that

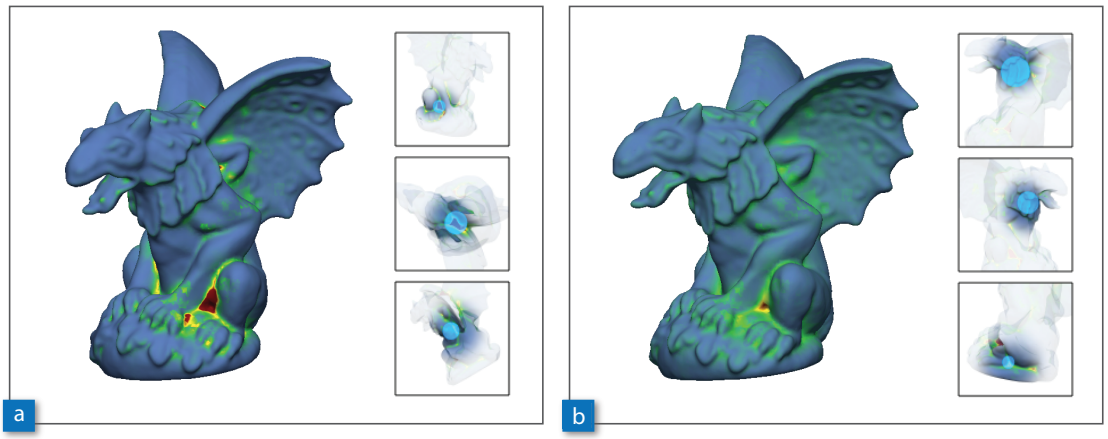

Figure 3.16: Changing the metric from point distances (a) to geometric deviations (b) results in an alternative overview image as well as different hot-spot locations.

additional information about the input data (e.g., noise level) would be helpful. We plan to integrate this into the system in the future. Berger et al. [BLN+13] implemented a surface reconstruction benchmark tool which can be used to test several reconstruction algorithms with one point cloud dataset. Additionally, the benchmark tool provides scanning simulations which can produce different point clouds of the same model with different quality (e.g., by adding more or less noise to the data). Then the benchmark tool can be used to apply the reconstruction algorithms to different point cloud versions. This would span a parameter space for every reconstruction algorithm showing its strengths and weaknesses (e.g., in the presence of noise).

During the evaluation, the domain experts also brought up another promising idea for future work. For them it would be very helpful to be able to **export** a 2D flattened or unrolled representation of the current model showing all, or at least the most important, hot-spots in one image, together with samples of the reconstructed meshes. Up to now such illustrations are generated manually by rotating the model and producing close-up views.

CHAPTER 4

Visual Analysis of Differences in Volume Ensembles

This chapter is based on the following publication:

Johanna Schmidt, Bernhard Fröhler, Reinhold Preiner, Johannes Kehrer, M. Eduard Gröller and Stefan Bruckner, and Christoph Heinzl. Visual Analysis of Volume Ensembles Based on Local Features. Technical Report TR-186-2-16-2, Institute of Computergraphics and Algorithms, TU Wien, Austria, 2016

Existing approaches for ensemble visualization already concentrated on how the multitude of information, and the variability of the data can be visualized in an effective and intuitive way. In this thesis, methods for visualizing image ensembles (Chapter 2) and 3D meshes (Chapter 3) have been discussed so far. The presented techniques demonstrated that many analysis tasks involve the exploration of local data regions where the ensemble members show slightly different characteristics. One major analysis task, which is also not covered by VAICo and the YMCA system yet, is that sometimes the local regions themselves need to be compared with each other. Otherwise it is particularly hard to tell whether a high data variance, or variability, is in all cases caused by the same, or by different ensemble members. It is also not possible to analyze whether a ensemble member producing reasonable results in one region of the data might be responsible for a high variance in another region. In YMCA, the parallel coordinates plot is a helpful took towards answering this question. Users can compare hot-spots regions by placing them close together in the plot, and track the performance of reconstructions algorithms by following the polylines. The plot, however, provides an abstraction of the underlying 3D data. If the users would like to explore the regions in 3D, they have to click on the thumbnails and move to a different view. The parallel coordinates plot of YMCA allows to track individual ensemble members across several local regions. Another approach from the weather simulation domain demonstrates how glyphs can be used to visualize similar regions at positions in a regular grid [JDKW15].

With such techniques it is not only possible to locate regions of interest in the data, but also to further compare these regions with each other. The existing approaches, however, do not offer the possibility to analyze how much individual ensemble members contribute to the local variance in a certain region. They are also only suited for the analysis of 3D meshes and 2D vector fields.

With the approach proposed in this chapter we show how local features can be analyzed in 3D volumetric ensemble datasets. We also show how the local regions can be further analyzed, and how this local information can be combined and summarized by using techniques from information visualization. As an exemplary use case for volume comparison, we demonstrate our techniques on ensembles of *volumetric segmentation masks*. Segmentation is applied to the volume data and based on the used parameter settings the resulting *segmentation masks* show slight differences. An overview of the data we used as an illustrative example can be seen in Figure 4.1. The analysis of the resulting segmentation masks helps to find out which regions of the volume data are critical to segment. Furthermore, parameter settings that lead to unwanted features in the data (e.g., noise) can be excluded from the working process in the future.

In our visual analysis approach, we first provide a general overview of the available ensemble data. For this purpose, we compute the *most representative* ensemble member that represents the data best. This most representative ensemble member is called the spatial median, or the *medoid*, of the ensemble. We compute the medoid by using the Weiszfeld algorithm [BS14]. The medoid is then used as an entry point for the local exploration. To guide users, we visualize areas of high variance in the data in 3D. These regions can be further explored using interactive *probing widgets*. The probing widgets consist of 3D spheres which can be positioned arbitrarily in 3D space by mouse interaction to investigate regions of high variance. While moving the widgets, the current local characteristics of the data are shown in a separate *detail view*. The detail view lists the outliers at the current position and gives insight into the underlying data. It is

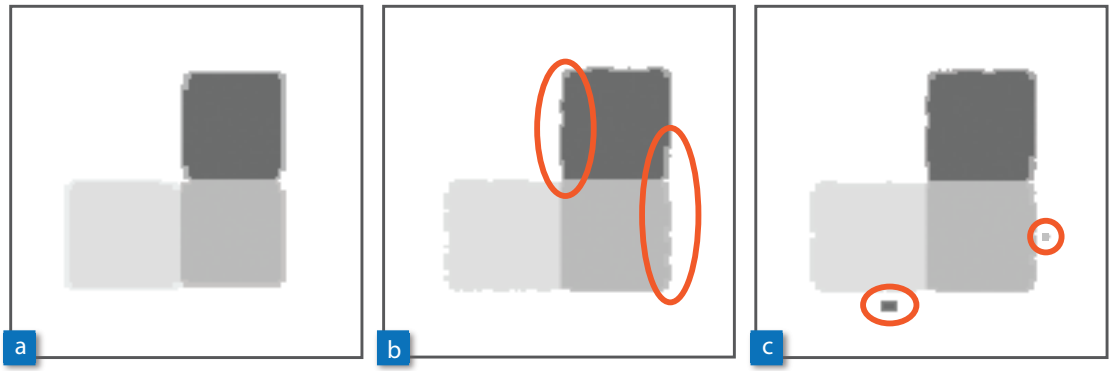

Figure 4.1: Different segmentation masks of the same volume dataset. The segmentation masks were created using different settings of the same parameter space. Optimal parameter settings result in a reasonable segmentation according to the analysis task (a). Different parameter settings may lead to under or oversegmentation, resulting in uneven surfaces (b), as well as incorporating or excluding regions due to noise in the data (c).

also possible to fix widgets at a user-defined position within the volumetric datasets. The fixed probing widgets are arranged in a *similarity graph* in 3D and 2D to indicate whether the local data exhibits comparable characteristics. In this way the users can explore local regions and analyze their similarities, without losing the context and the original data.

The main contributions of our approach are:

- *Volume probing*: Our proposed interactive probing widgets allow users to explore local regions in volumetric ensemble datasets. We provide means to place probes inside the volume to specify regions of interests.
- *Similarity graph*: The spatial position and extent of the probing widgets are used to arrange the local regions in a graph according to the similarity of the local data. The graph data can be shown in 3D and in 2D.
- *Multi-level analysis*: With the similarity graph it is possible to analyze the whole ensemble data and compare local regions of interest. It is also possible to compare one or more ensemble members against the rest of the ensemble.

4.1 Related Work

The approach proposed in this chapter is strongly related to location-based ensemble visualization, as introduced in Section 1.2.2. We use interaction tools in 3D, and we extend the ideas of Jarema et al. [JDKW15] to the field of volumetric datasets, where widgets also need to be placed inside the data. We also make use of local **volume probing** to select regions of interest, which has already been introduced in Section 1.2.3.

As an exemplary use case, we analyze collections of volumetric **segmentation masks** in this paper. Segmentation algorithms typically use several parameters which require careful tuning. *Tuner*, as proposed by Torsney-Weir et al. [TWSM+11], allows user to analyze different segmentations of 2D images. Tuner requires the definition of at least one objective quality measure, for example, defined through the difference to a ground-truth image, and concentrates on investigating these derived measures instead of the segmentation masks. Geurts et al. [GSK+15] proposed a system to analyze different segmentations of medical data. They also require objective quality measures and provide local analysis methods based on these measures. The *GEMSe* tool by Fröhler et al. [FHM16] can be used to browse an ensemble of 3D segmentation masks to identify parameter settings for a proper segmentation result. It does not require an objective quality measure, but provides limited support for local exploration. While segmentation exploration systems mainly focus on browsing the full dataset, our technique allows users the detailed exploration and comparison of local features. This is useful to analyze the slight differences of segmentation masks in a subset of the ensemble, where the ensemble members have been previously defined as useful. Our work can therefore be seen as an extension to the work by Fröhler et al. [FHM16] and the work by Geurts et al. [GSK+15], improving these with respect to the local analysis. Our technique also operates on datasets without a proper reference solution.

4.2 Visual Analysis of Volume Ensembles Based on Local Features

Our visual analysis approach for the comparison of volume ensembles allows for the inspection and comparison of local regions in the data. As an exemplary use case, we focus on the comparison of volumetric segmentation masks. The masks have been created by segmenting a specimen with the same segmentation algorithm, using different parameter settings. The segmentations are created from data from different domains. We do not have ground-truth data available in the ensembles. Our comparison approach is, in particular, designed to compare the statistical properties of local regions in the data. The proposed visualization techniques can further be used to compare individual, or groups of ensemble members, against the rest of the ensemble.

To be able to inform users about the differences in the ensemble, the individual ensemble members need to be compared with each other, which is described in Section 4.2.1. The comparison involves the computation of variances in the data, and distances between the ensemble members. We further need structures to present the differences to the users in a 3D environment. For this we identify the most representative ensemble member using the Weiszfeld algorithm, as described in Section 4.2.2. This ensemble member is then used to display the local differences in a 3D environment. The interaction possibilities are described in Section 4.2.3. To be able to compare local regions in the data, we arrange them in a graph. The construction of the graph is outlines in Section 4.2.4. An overview of the pipeline of our approach is given in Figure 4.2.

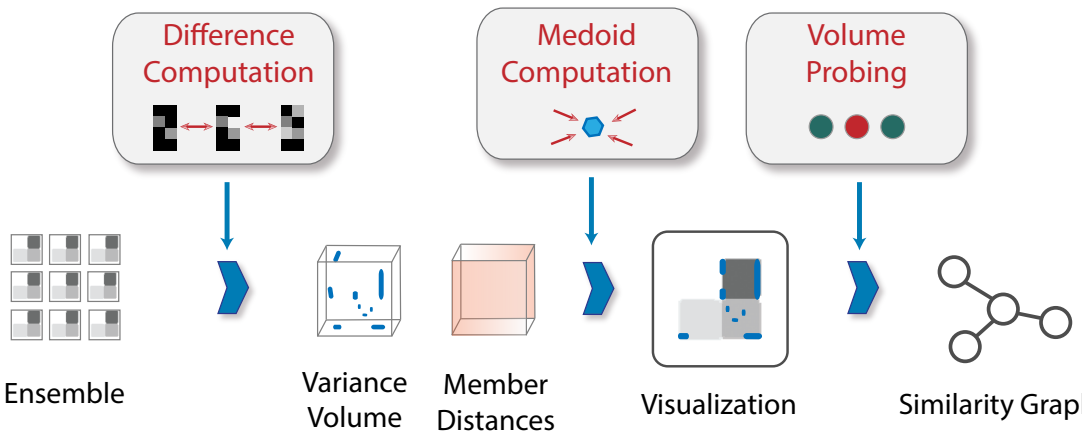

Figure 4.2: Overview of our visual analysis approach. We start with an *ensemble* of volumetric segmentation masks. After the *difference computation*, we obtain the *variance* and the *distances* of the ensemble members in the data. This data is then presented to the users in a *3D visualization* by showing the most representative ensemble member. There users can use *volume probing* to explore local regions. To compare these regions, they are arranged in a *similarity graph*.

4.2.1 Pre-processing and Difference Computation

Before being able to start the visual analysis of the ensemble data, the ensemble members need to be compared with each other. The input data consists of an ensemble of volumetric segmentation masks, and we do not have ground-truth data available. We are especially interested in locating regions of high variance in the data, since these are the regions where the segmentations produced different results.

All of the following calculations are not computed on the raw input volume data, but on their **distance transforms**. In a distance transform, or distance map, for each voxel in a volume the distance to the nearest *obstacle voxel* is assigned [ST94]. In our case, obstacle voxels are defined as the voxels representing the segmented surface. In Figure 4.3 a volumetric segmentation mask and its corresponding distance map is shown. We use distance maps instead of the raw data for further comparison, as they provide a more robust measure than comparing the raw pixel values. Therefore, in a pre-processing step, a distance map is created for every input volume. The created distance maps are of the same dimension and spacing as the input volumes.

In the next step additional parameters are extracted from the distance map data. For this purpose we create a new volume of the same dimensions as the distance maps. Then, for every voxel v, the mean value $mean_v$ is computed for all voxel values $value_{n,v}$ in all n ensemble members. The per-voxel mean values are stored in a newly created **mean distance map**. The mean values are needed in the next step where we calculate the local variances in the dataset.

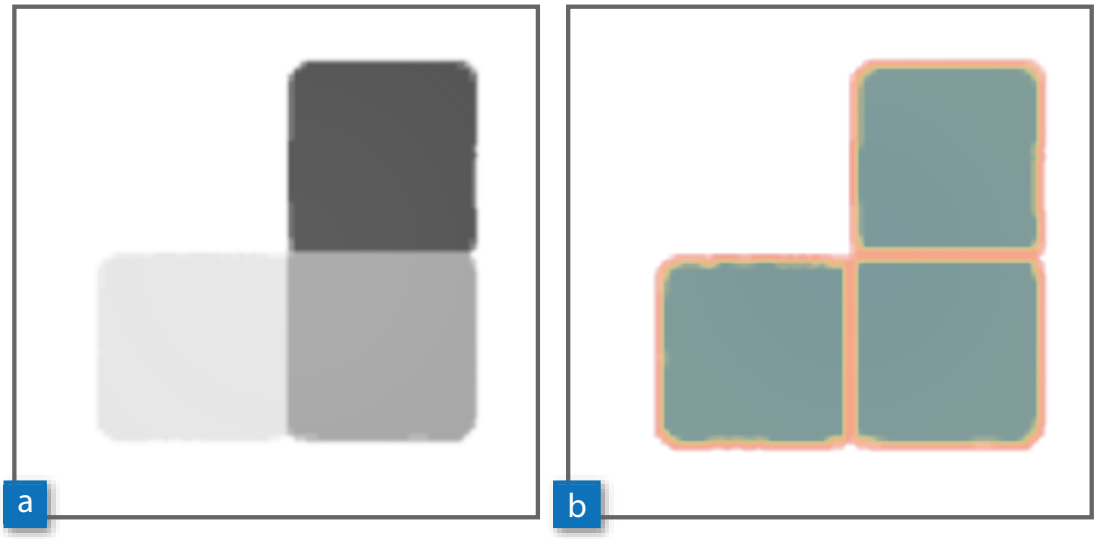

Figure 4.3: Distance maps. This figure shows a volumetric segmentation mask (a) and its corresponding distance map (b). The distances from the surfaces are highlighted by colors from blue (further away) to orange (close). The distance map stores the per-voxel distances to the respective nearest *obstacle voxel*. The obstacle voxels are in this case defined by the voxels representing the segmented surface.

The variance is a very useful tool to measure the variability of a given set. We again compute the variance in a per-voxel process. For every voxel v, the variance σ_v^2 is calculated with the standard formula as follows:

$$\sigma_v^2 = \frac{\sum(mean_v - value_{n,v})^2}{n} \tag{4.1}$$

The per-voxel variance values can be combined into a new volume, which then defines the **variance distance map**. During the computations we also store the overall mean variance mv, which is a single value computed as follows:

$$mv = \frac{\sum \sigma_v^2}{k} \tag{4.2}$$

The paramater k defines the number of voxels, namely the dimension of the ensemble members, computed as $k = width * height * depth$.

In addition to the variance, we would like to keep track of which ensemble members are responsible for the high variance in a certain region. We therefore also store the total distance of one ensemble member to the other ensemble members in every voxel v. This distance $dist_{j,v}$, with $1 \leq j \leq n$, is again computed in a per-voxel process, and is defined as the sum of all $n - 1$ mean-squared Euclidean distances to the other ensemble members' voxel values:

$$dist_{j,v} = \sum_{i \neq j}^{n} (value_{i,v} - value_{j,v})^2 \tag{4.3}$$

We store the per-voxel distances in a new **ensemble member distances volume**. The mean distance map can be deleted after the variance computation is finished to save memory. The variance distance map and the ensemble member distances volume are kept to be later used in the visual exploration (Section 4.2.3).

4.2.2 Medoid Computation

We need a starting point in the visualization, from where the users can start the visual exploration of the local regions. Since we follow the principle of *overview first* and *details on demand*, we need to provide an overview to the users. The overview should convey how the ensemble data looks like (i.e., the phenomenon described by the ensemble). There are a lot of possibilities to present ensemble data in an aggregated way, for example, by simply calculating the mean ensemble member [JSW05]. Using an actual ensemble member, though, is generally favorable over a synthetic representative such as the mean [SPA+14]. Picking a random ensemble member might give a wrong impression of the actual data, because an ensemble member with a lot of artifacts might be picked accidentally. We therefore decided to calculate the most representative ensemble member, which we then call the *medoid ensemble member*.

Figure 4.4: Ensemble medoid. The input volumes can be interpreted as points in a k-dimensional space (*blue*). The mean (m) is usually prone to follow outliers in the dataset, and the nearest neighbor (*arrow*) of the mean would not be a good representative of the dataset in this case. The spatial median (sm) minimizes the Euclidean distances to all points, but is generally not a part of the dataset. We therefore find the point that is closest to the spatial median (*arrow*) and use this as the medoid (med) of the ensemble.

Apart from the fact that the mean is not part of the ensemble, it is also known to be sensitive to outliers. Therefore, it may happen that the mean is actually located far from the cluster of real samples. This is shown in Figure 4.4 for points in a 2D environment, where the presence of a single significant outlier (right) creates a mean (m, in orange) that is a blend between two clusters that does not relate to any of the ensemble members. Therefore, the mean would not be a good representation of the dataset. The figure also demonstrates that a mean-based medoid (i.e., the nearest neighbor of the mean in the ensemble) can also result in a bad representative object. Therefore, we choose a medoid based on the *spatial median* of the dataset (sm, in green). By interpreting each sample volume as a point in k-dimensional-space, its spatial median sm is defined as the k-dimensional point x that minimizes the sum of Euclidean distances to all other n points pt_n as follows:

$$sm = \underset{x}{\operatorname{argmin}} \sum_{n} ||x - pt_n|| \tag{4.4}$$

Equation (4.4) can be solved using the Weiszfeld algorithm [BS14], which, starting from the mean, approximates the minimum using the following fixed-point iteration:

$$x = \frac{\sum_n pt_n \, ||x - pt_n||^{-1}}{\sum_n ||x - pt_n||^{-1}} \tag{4.5}$$

In the 1D case, the median is always an ensemble member. This is, however, not the case for the spatial median in nD ($n > 1$). The spatial median is generally not part of the dataset, and can therefore not directly be used as the medoid of the ensemble. Therefore, we instead choose the medoid as the k-dimensional Euclidean nearest neighbor of the resulting spatial median. This is also illustrated in Figure 4.4 (med, in blue).

In the following step, this **ensemble medoid** will be used to represent the dataset, and to start the visual exploration of the local regions. The calculation of the medoid only has to be done once when loading the ensemble into the system.

4.2.3 Local Exploration in 3D

Our proposed visual analysis technique provides different views onto the data, which can be influenced by different interaction techniques. The concept of our interaction techniques is described in Figure 4.5. The individual elements of the concept are described in this section.

The interaction works mainly in 3D, and therefore the most important view is considered to be the **3D view**. As a starting point for the exploration, we display the medoid volume as a representation of the ensemble dataset in this view. In addition, the regions of high variance are also shown in 3D. The per-voxel variance values have been stored in a separate variance distance map. We decided against standard volume rendering to display the variance values, to make a clear distinction between the medoid and the rendered variance values. To show the regions of high variance, we use Marching Cubes [LC87] to transform the variance values into mesh data.

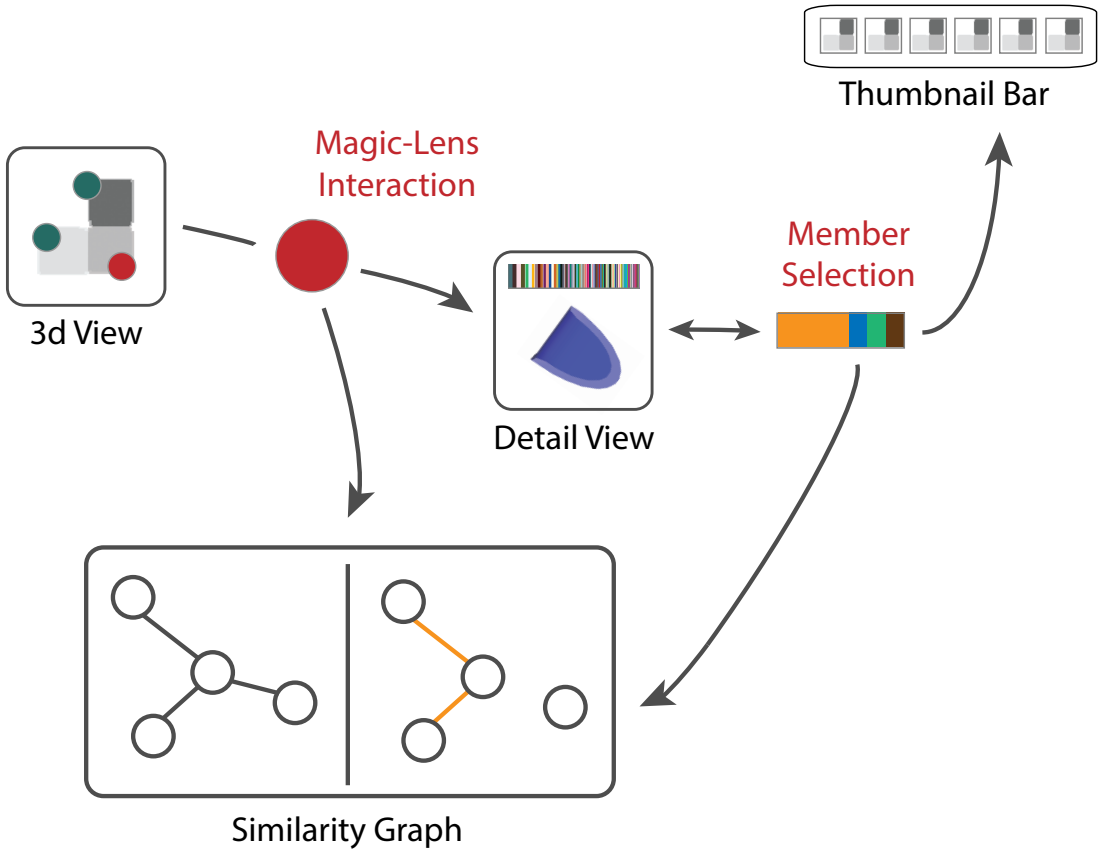

Figure 4.5: Interaction concept. Starting from the 3D view, the user can employ *volume probing* to explore local regions. The current position of the probe influences the *detail view* and the *similarity graph*. The *member selection* possibility in the detail view has an impact on several other views. The similarity graph has two stages, either for comparing the whole ensemble data, or just comparing one ensemble member against the rest.

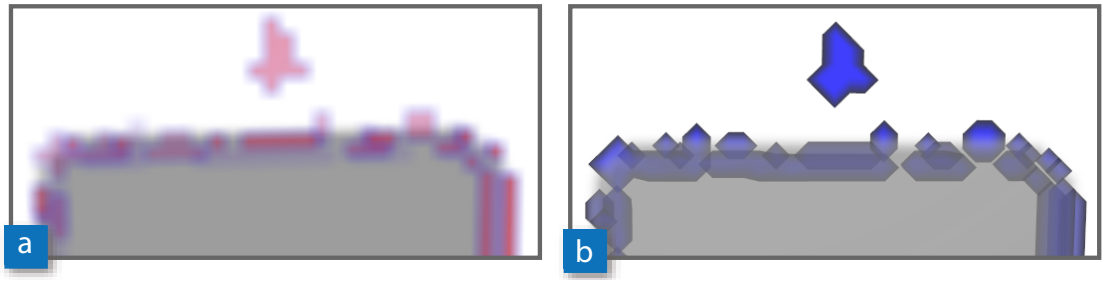

Figure 4.6: Regions of high variance in 3D. For displaying high variance in the data, we decided against volume rendering (a) to make a clear distinction between the variance information and the medoid volume (b).

The threshold for the Marching Cubes algorithm is set to divide between regions of *low* and *high* variance. We used the mean variance mv (as computed in Section 4.2.1) as the input threshold for the Marching Cubes algorithm. All values above this threshold are considered to represent a high variance in the data. The mean variance is used, instead of an outlier computation like the Z-score, to capture as many areas with a high variance as possible. This way geometry is created for regions of high variance, and regions of low variance are discarded from the visual representation. Since we operate on mesh data now, it is possible to render it in a way that it is visually easily discriminable from the medoid volume (Figure 4.6).

In the 3D view users can explore local regions by using our proposed **volume probing**. When activated by a hotkey, a probing widget follows the position of the mouse cursor. Users can also control the movement in the third dimension by keeping the hotkey pressed and using the mouse wheel. Then a clipping plane will be shown and moved along the z-axis of the volume, and the current probing widget will always be positioned on this plane. Users can employ the probing widgets to explore the regions of high variance. We opted for using 3D spheres as probing widget due to their intuitiveness and rotation-invariance. The size of the spheres can be individually adjusted by the users. Furthermore, users can fix widgets at the current position by mouse click. The widget is then fixed and stored in a sorted list, which is visible next to the 3D view. In this list users can activate fixed widgets again by mouse click.

An important visual channel to convey information in our technique is color. If new ensemble members are added to the ensemble, we assign a unique color to each of them that stays the same for the time our tool is running. We call these colors the **ensemble member colors**. Whenever a ensemble member is selected in the interface, it is important that information related to this ensemble member is visible in other views. We decided to indicate these relations with color, and therefore a unique color is assigned to every ensemble member. We created a color table of 100 colors by using the *i want hue* tool [Jac16]. Whenever an ensemble member is added to the analysis, a new color is selected for it from the table. It is also clear that not more than approximately 10-15 colors can be discriminated in one view. It has to be mentioned that when selecting colors to relate information in the views, it is not the purpose of our technique to compare a multitude of ensemble members (i.e., more than 10-15) against each other.

The position of the currently active probing widget is used to update the **detail view**. The detail view is placed next to the 3D view and shows the data characteristics at the current probe position. On the top a stacked bar-chart with all ensemble members is shown (Figure 4.7a). As described in Section 4.2.1, we store the Euclidean distances of the ensemble members in a separate volume. At the local probe position, we now iterate through all voxels covered by the probe and sum the distances $dist_v$ for all ensemble members. The summed values are then displayed in the bar-chart. The larger a segment in the stacked bar, the larger the distance of the ensemble member to all others. A large segment thus indicates here that the corresponding ensemble member differs from the rest of the ensemble members in the current local region. Below the chart, a more detailed view on the local data is presented. This view reveals more information about the data (e.g., statistical properties) which is not visible in the 3D view. We decided to show a 3D rendering of the segmented surface of the medoid (Figure 4.7b). Since we are dealing with segmentation masks, an additional information we can present to the user (apart from the volume rendering of the medoid) is the shape of the segmentation surface at the current local position. The local differences in the data are shown as 3D structures in the rendering. If individual ensemble members have been selected (this is also explained below), their surfaces are shown next to the medoid surface (Figure 4.7c). The viewing direction of the surface rendering in the detail view is adapted according to the main viewing direction in the 3D view.

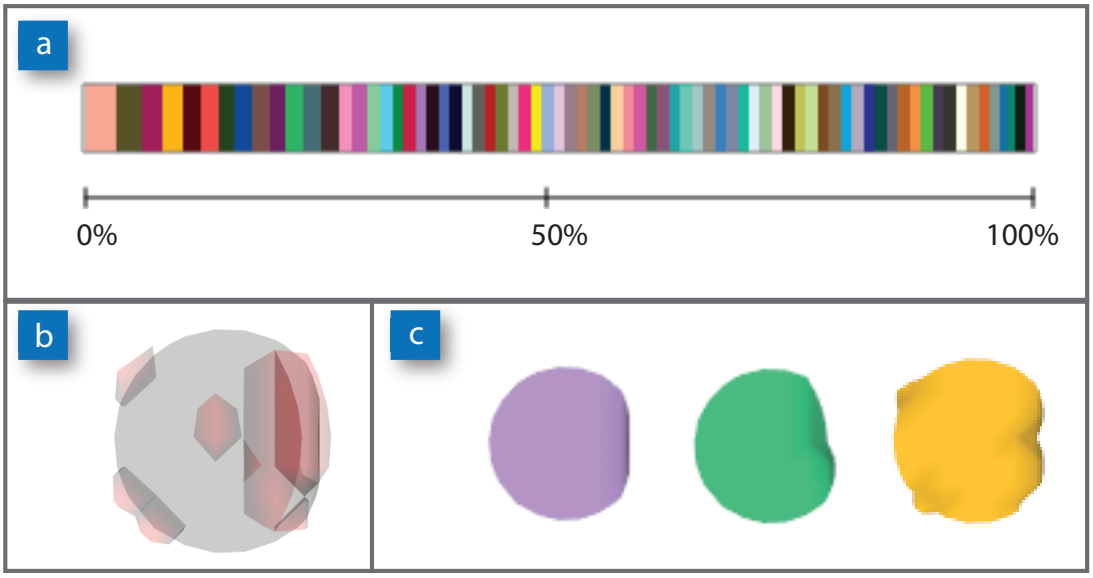

Figure 4.7: Detail view. This view is updated when the currently activated probing widget is moved. The detail view shows a stacked bar-chart to indicate the distances of the ensemble members at the current position (a). Below a rendering of the cropped surface extracted from the medoid can be seen (b - minimized). The surface is sphere-shaped, since it has been cropped at the current probe position. The high variance regions are indicated on the medoid surface in red color. In the area below, surfaces of ensemble members can be shown next to each other (c).

In the detail view, the **member selection** tools can be used. It is possible to select individual ensemble members in the bar-chart by clicking in the rectangles in the stacked bar. This ensemble member selection triggers other views to update, because it has a strong influence on the visualization. First of all, a member selection updates the detail view itself, because if ensemble members are selected, their surfaces are shown in the detail view together with the medoid surface. This way the data of several ensemble members can be compared to each other.

In our user interface, we also preserve the view on the individual ensemble members. The input volumes are represented as thumbnails in a **thumbnail bar** above the other views. If hovering over the thumbnail, the filename of the ensemble member is shown. The medoid is marked by a grey background in the thumbnail. Users can open a 3D view of the original segmentation mask volume data by a mouse click on the thumbnails. Members which have been previously selected in the detail view are marked with a colored border. The thumbnail view allows users to always go back to the original data, if necessary.

4.2.4 Similarity Graph

An important contribution of our technique is the indication of the similarity between different probes. This allows users to compare local regions, and to find similar or distinctive regions in the data. The users can fix the position of the probes, and all fixed probes are then arranged in a **similarity graph**. The graph is shown in the interface in 3D and 2D. The 2D representations reveals patterns in the graph, while the 3D representation shows the spatial positions of the nodes.

In the graph, nodes represent probes and edges indicate their similarity. If two probes are identified to be similar, they are connected by an edge in the similarity graph. The edges are colored accordingly to identify patterns in the graph. The similarity of two probes is computed by analyzing the local outliers at the probe positions. We make use of the sum of distances for every ensemble member, which is automatically computed for every probe position. We analyze these sum of distances using statistical methods to identify outliers. First, we compute the median $median_p$ of all sums of distances for the current probe p with the standard formula. Then we compute the median absolute deviation [How14] for the probe, which is a robust estimate for the standard deviation. The median absolute deviation mad_p is computed as follows:

$$mad_p = 1.4826 \cdot median(|dist_{p,n} - median_p|) \tag{4.6}$$

A modified version of the Z-score, as proposed by Iglewicz and Hoaglin [IH93], can be used to compute an outlier index $o_{p,n}$ for the current probe as follows:

$$o_{p,n} = |\frac{dist_{p,n} - median_p}{mad_p}| \tag{4.7}$$

An ensemble member is considered to be an outlier in a voxel v if $o_{p,n} >= 3.5$ holds [IH93]. In the case mad_p is equal to zero, the equation cannot be solve. Then every distance deviating from the median is considered to be an outlier. After the computation, we end up with a list of

$1 \ldots n$ binary flags for every ensemble member n inside the probe. The flags indicate whether an ensemble member has been identified as an outlier ($flag = 1$) for the current probe position or not ($flag = 0$). For all fixed probes the flags can be compared to compute the similarity.

The nodes of the graph are always colored in the same color, since they refer to the probes in the 3D view. The way edges are colored in the graph, and how the similarity is computed, depends on the current *stage* of the graph. The different stages allow users to explore different aspects of the data. The following three graph stages are possible (see also Figure 4.8):

- **Analyze all**: *No members have been selected.* In this case the whole ensemble data can be analyzed. The similarity is computed in a way that if all outlier flags for all members are equal, two probes are considered to be similar. The graph edges are all displayed in the same color.
- **Analyze one member**: *One member has been selected.* In this case one member can be compared against the rest of the ensemble. The similarity is computed in a way that only the flags for this member are considered (if the member is a local outlier) - in case the flags are equal, two probes are considered to be similar. The graph edges are colored in the member color, if the member has been identified as an outlier in the probes (no edges are drawn otherwise).
- **Analyze a group of members**: *Between* 2 *and* $n - 1$ *members have been selected.* In this case several members can be compared against the rest of the ensemble. The similarity is computed in a way that only the flags for the selected members are considered (if the members are local outliers) - in case the flags are equal, two probes are considered to be equal. The graph edges are colored in the member color of the members that have been identified as outliers.

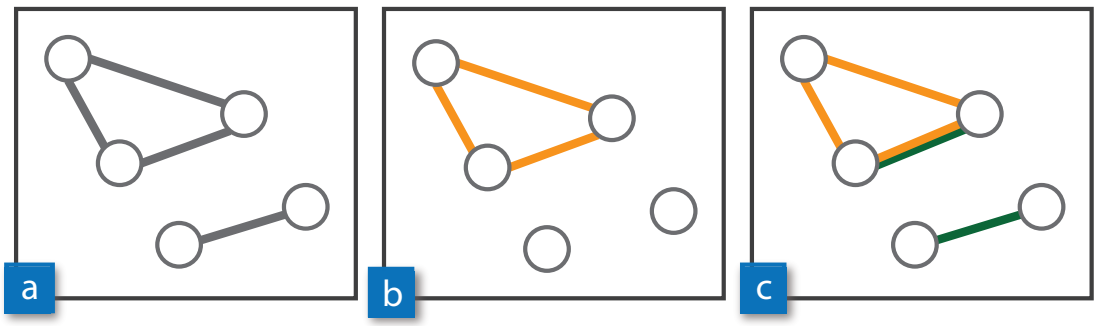

Figure 4.8: Similarity graph. The visualization of the graph depends on its stage. If the whole ensemble is analyzed, the similarity is indicated by lines (a). If only one ensemble member is compared against the rest of the ensemble, the edges between probes where it has been identified as an outlier are colored in the ensemble member color (b). If more than one ensemble member is compared against the rest of the ensemble, the edge colors indicate which ensemble members have been identified as outliers at the respective positions (c).

4.3 Implementation

Our technique was implemented in C++, and the user interface was implemented by using the Qt[1] framework (version 5.5.1 64bit). Loading and rendering of the volume data was done using the VTK[2] framework (version 7.0.0 64bit). VTK was also used to compute the distance transforms of the volumes, to render the 3D elements and the similarity graph, and for handling the user interactions. The application was tested on an Intel i7 CPU 3.07 GHz machine with 24 GB RAM and an NVIDIA GeForce GTX 680 graphics card.

To analyze the data, pre-processing and the computation of the differences as well as the outliers is necessary (see Section 4.2.1). This step is not interactive, so the user has to wait until all attributes are calculated. It is however possible to cache intermediate steps (e.g., the computed distance transform, the medoid, and the variance outlier volume), so that a repeated load of the same dataset is faster. A detailed description of the datasets used for the demonstration can be found in Section 4.4.

Table 4.1: Runtime and memory requirements. In this table the dataset dimensions and number of ensemble members, the runtime for the analysis (for computing the dataset differences and the medoid), and the memory requirements are shown.

Ensemble	**Dimension**	**Members**	**Analysis**	**Memory**
Synthetic	120x120x8	74	47 s	1.1 GB
Rock Crystal	285x300x216	51	2.6 m	14.2 GB
Lower Body	512x512x801	13	5.3 m	16.9 GB
Aorta	512x512x805	21	6.1 m	17.3 GB

An overview of the runtime of the preprocessing steps for each dataset is shown in the Table 4.1. The column *Analysis* shows the time it took to compute the distance maps, the volume differences, and the medoid for every dataset. The computation of the variance and the distances between the ensemble members could be greatly speed up by distributing the computations into several threads. Since the variance and the distances are computed in a per-voxel process, the computations are completely independent and can therefore run in parallel. The column *Memory* shows the overall memory consumption during runtime. More discussions on the runtime and the memory consumption can be found in Section 4.5.

[1] http://www.qt.io/
[2] http://www.vtk.org/

4.4 Results

We applied our technique to volumetric segmentation mask ensembles from different domains. All datasets have in common that all ensemble members are of the same size. Since all input volumes are of the same dimension and spacing, we can rely on the fact that every voxel position is present in every volume, and also represents the same spatial position in every volume.

4.4.1 Segmentation of Industrial-CT Data

The first two datasets we tested our approach with came from the domain of *industrial computed tomography (CT) analysis*. The segmentation masks were derived from industrial CT datasets by applying the extended random walker algorithm [Gra05] to segment the data. The algorithm was run several times with different parameters, which created ensembles of segmentation runs.

The dataset *Synthetic* consists of 74 different segmentation results of an artificially created industrial workpiece with quite small dimensions (120x120x8). In this dataset we could identify interesting artifacts outside the boundaries of the actual object. Such artifacts usually occur if a segmentation algorithm is affected by a lot of noise during the calculation. This may happen due to the inherent structure of the algorithm, or due to improper parameter settings. It is therefore important to know which ensemble members are responsible for such artifacts in the data.

With our technique artifacts can be analyzed by placing probing widgets at the positions of the artifacts that should be analyzed. One placed, the stacked bar in the detail view clearly indicated that in all cases only one ensemble member produced these artifacts. We then fixed the probing widgets at the positions of interest. The probes were then connected by edges in the similarity graph, which means that they are placed in regions with similar data. Figure 4.9 shows the similarity graph in 3D (Figure 4.9a) and some of the stacked bars of the detail views of the fixed probes (Figure 4.9b). In the Figure it can also be seen that one of the probing widgets is colored in red (rightmost one). This probe has not been fixed yet, so its position and size can still interactively be changed. Nevertheless, the probe is already included in the similarity graph. This way the active probe can be used to interactively compare new regions to the already fixed ones. If the region the active probing widgets is currently placed at is similar to the fixed ones, the probes are connected in the graph (as it can be seen in Figure 4.9). If the region the active probe is currently placed would not be similar, there would be not connecting edges between the fixed ones and the active probe. This way users immediately recognize if they are currently moving into region which is similar to data already stored in the similarity graph. This way users can check whether a high local variances are always caused by the same ensemble member or by the same group of ensemble members.

The dataset *Rock Crystal* represents a piece of a rock crystal, which has been scanned by an industrial CT scanner. The dataset consists of 51 ensemble members with 285x300x216 voxels each. The segmentation of the inner structures is a non-trivial task, and therefore a lot of regions of high variance could be located inside the object (Figure 4.10a). Some other artifacts are also visible outside the rock crystal. These artifacts appear because some segmentations included parts of the scanning equipment which were also present in the scanned data.

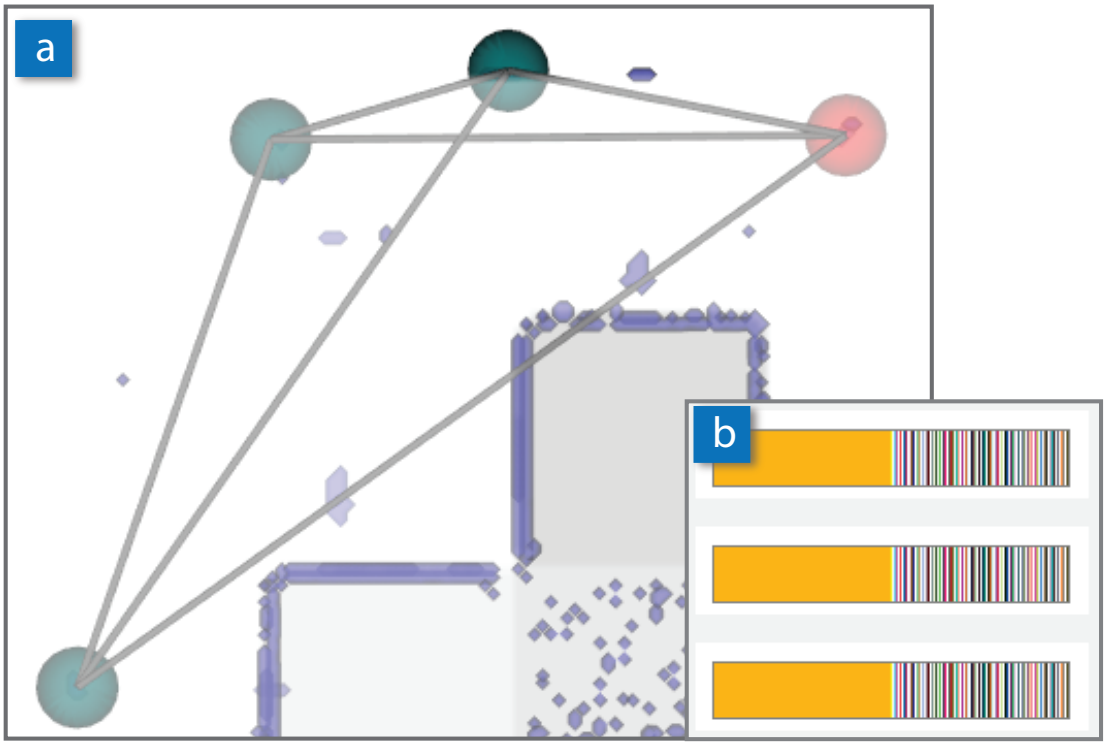

Figure 4.9: Dataset *Synthetic*. When analyzing the artifacts outside the object, the connected nodes in the similarity graph show that the local distribution of the differences is similar (a). In the detail view, it can be seen that only one ensemble member causes the artifacts in the ensemble (b). The probing widget in red is currently active and can be moved by mouse interaction.

To analyze these artifacts, we placed several probing widgets inside the object, and also at the artifact regions in the outer regions. The fixed probes can be further analyzed in the similarity graph. First we had a look at the graph with no ensemble member being selected (i.e., stage *Analyze all*). There we could identify two clusters in the graph, in 2D as well as in 3D (Figure 4.10b). This means that both the regions of the outer artifacts and the inner structures represent similar data. The 2D representation of the graph allows to detect patterns in the graph, whereas the 3D representation better reveals the spatial position and size of the nodes. We could identify one ensemble member that produced different results inside the rock object than the rest of the ensemble. We selected this ensemble member in the detail view. The similarity graph then changed (i.e., stage *Analyze one member*), and the outer probes are no longer connected in the graph (Figure 4.10c). Only the larger connected group of nodes remains in the graph. This shows that the selected ensemble member produces different results inside the rock object, but is not responsible for the artifacts in the outer regions. The ensemble member is therefore responsible for the high variance inside the rock crystal, but is not responsible for the outer artifacts.

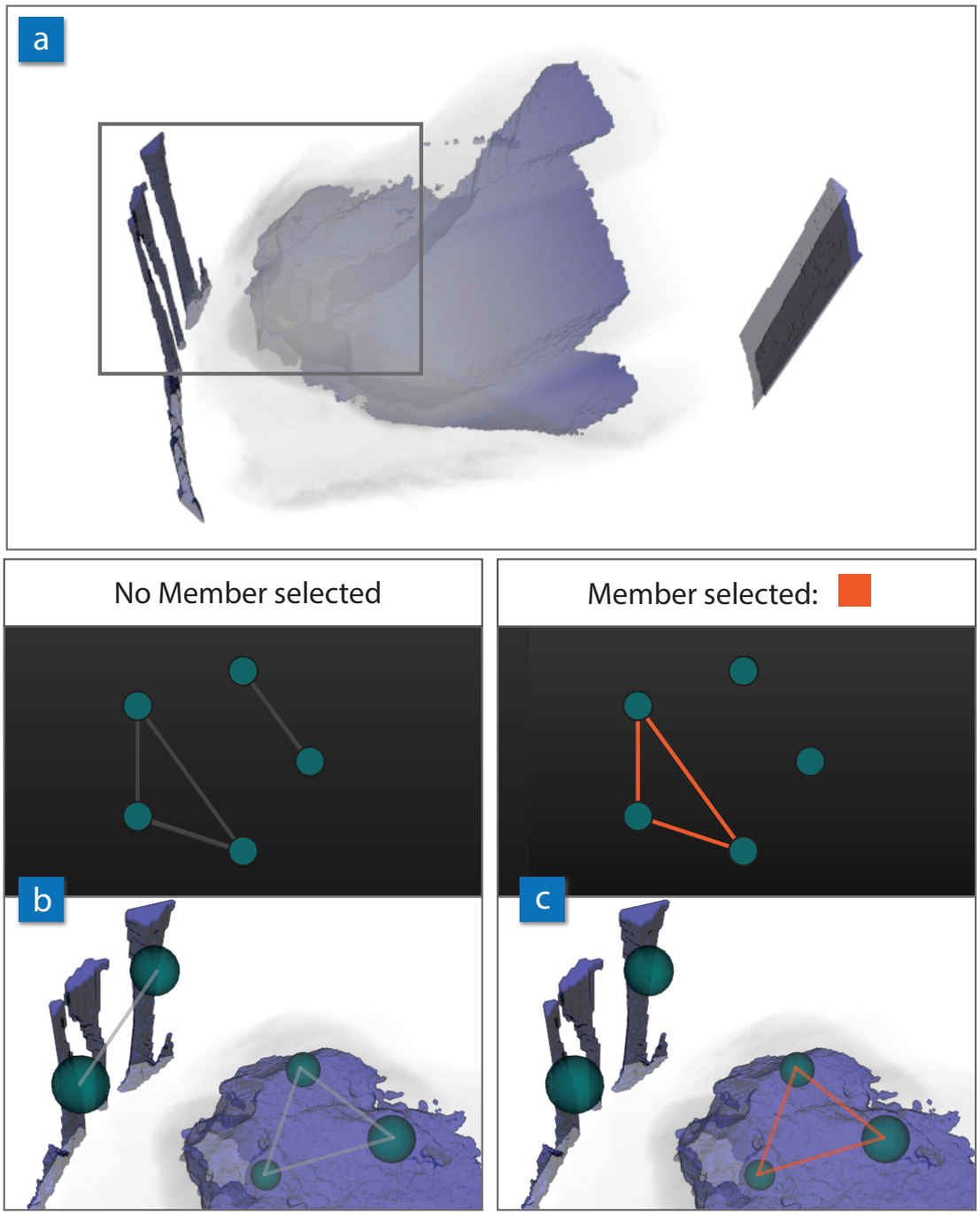

Figure 4.10: Dataset *Rock Crystal*. This dataset contains results of segmentations of a rock crystal. A view on the complete medoid with the rendered variance regions can be seen in (a). In a close-up of the data (grey rectangle), we placed probing widgets to analyze the local data. Similar regions could be found, which were then connected in the similarity graph (b). When selecting one ensemble member, it can be seen in the graph that this ensemble member only produces different results inside the object (c), but is not responsible for the outer artifacts.

4.4.2 Segmentation of Medical Data

We applied our technique to two datasets from the *medical domain*. Both datasets were generated by computed tomography angiography (CTA), and represented segmentations of inner human structures, like bones and vessels.

The first medical segmentation dataset, called *Lower Body*, was created to segment bones and vessels in the lower body of one patient Figure 4.11a. The data was segmented using the *AngioVis ToolBox* software [Str06]. Different levels of Gaussian noise were added to the data, to analyze the impact on the segmentation algorithm. One ensemble member of the ensemble therefore corresponds to a certan noise level in the data. Our technique can be used to compare the noise levels, to see which ones affect the segmentation, and which do not. The ensemble consisted of 13 ensemble members with dimensions of 512x512x801 voxels each. First, the regions of interest for the analysis can be defined by placing the probing widgets accordingly. In Figure 4.11b and Figure 4.11c three of the placed widgets can be seen. Selecting an ensemble member in this case means comparing the results of a segmentation with a certain noise level to the rest of the ensemble. In the first case (Figure 4.11b) the ensemble member produces results that are significantly different from the rest of the ensemble. The nodes are therefore connected in the similarity graph. This means that this noise level greatly effects the outcome of the segmentation. In the second case (Figure 4.11c) the ensemble member produces similar results like the other algorithms in the ensemble. This can be seen in the similarity graph, since the nodes are not connected (i.e., stage *Analyze one member*). This noise level therefore does not affect the segmentation results. This demonstration case shows that our technique can be used to quickly verify whether a segmentation is affected by certain factors. It also allows users to concentrate on only the regions of interest that are important for their analysis.

The second dataset, *Aorta*, was created to segment the different vessel segments in case of an aortic dissection. An aortic dissection is a malformation of a blood vessel, caused by a tear of the inner vessel wall. Successively, blood flows between the diverged layers of the arterial vessel wall, leading to the creation of two blood flow channels instead of one. The two channels are called the *true lumen* (TL), the channel where blood is still flowing, and the *false lumen* (FL), the channel where the blood flow has stopped. At slices perpendicular to the centerline a 2D level set segmentation [CV01] was applied to segment the blood flow channels. The per-slice 2D segmentations were then combined into a 3D volumetric segmentation mask again. Due to a variety of true and false lumen configurations, a perfect segmentation cannot be expected. A perfect segmentation would separate the TL and FL, if available, in the slice. If the TL and the FL are very close together, it may happen that the TL segmentation flows over to the FL area. If other structures with a similar intensity value like the aorta (e.g., bones) are close to the aorta, the segmentation may flow over into these unwanted parts. The dataset consisted of 21 ensemble members with dimensions of 512x512x805 voxels. The ensemble members of the ensemble have been created from the same patient data, with slightly different parameter settings for the level-set segmentation. The segmentation was done for the full aorta, however, our collaboration partners pointed out that they are actually only interested in the lower parts of the aorta. Since our technique operates on local regions, it was possible to concentrate on only the regions of interest. Segmentation in the medical domain is a common problem, and suitable solutions that would work

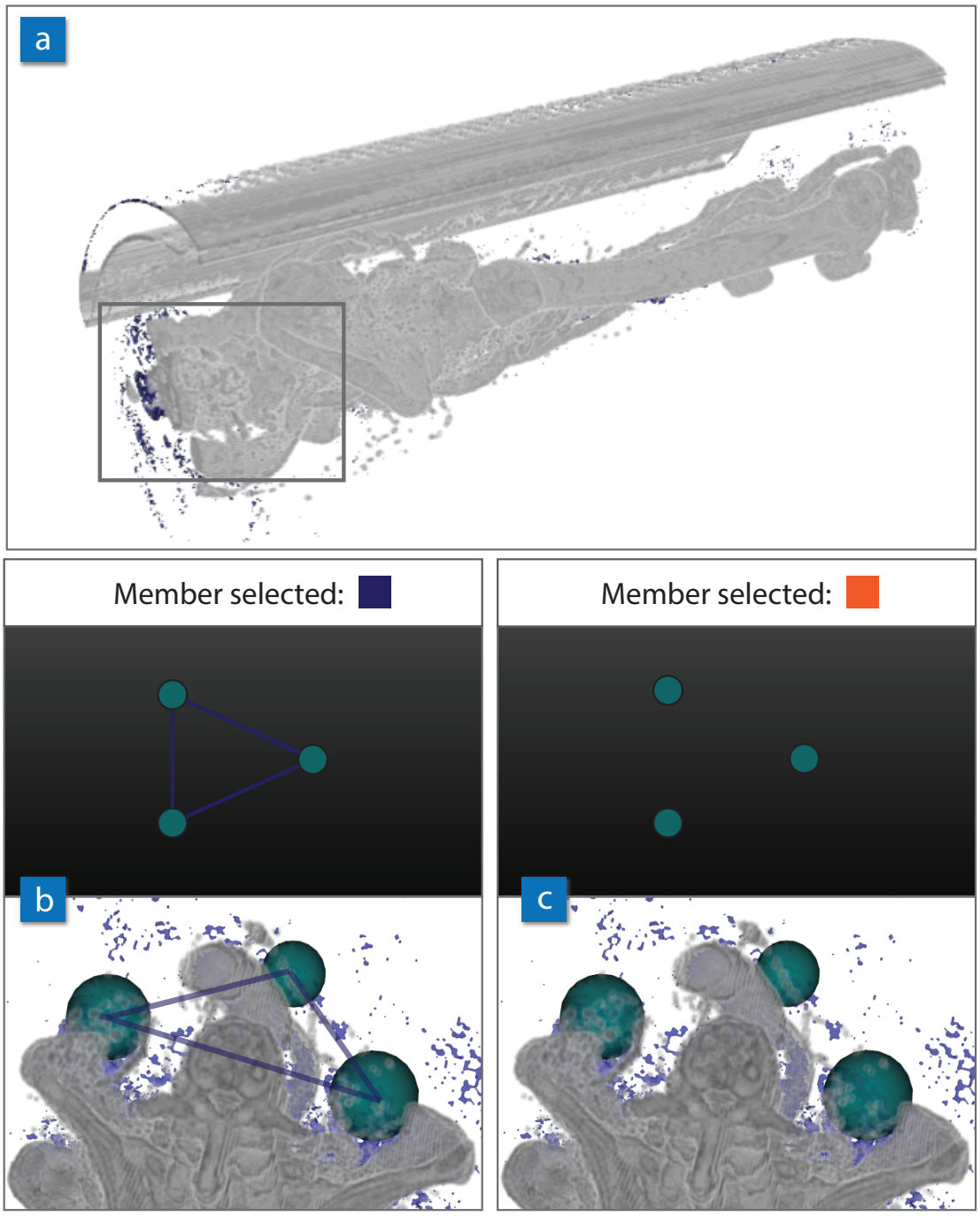

Figure 4.11: Dataset *Lower Body*. In this dataset bones and vessels of a patient's lower body have been segmented (a). The ensemble members also refer to different noise levels in the data. In the close-up (grey rectangle) three probes have been placed. In the first case (b) the ensemble member greatly effects the segmentation, because here the results are significantly different than the rest. In the second case (c), the results do not differ from the existing data, and therefore the nodes in the graph are not connected.

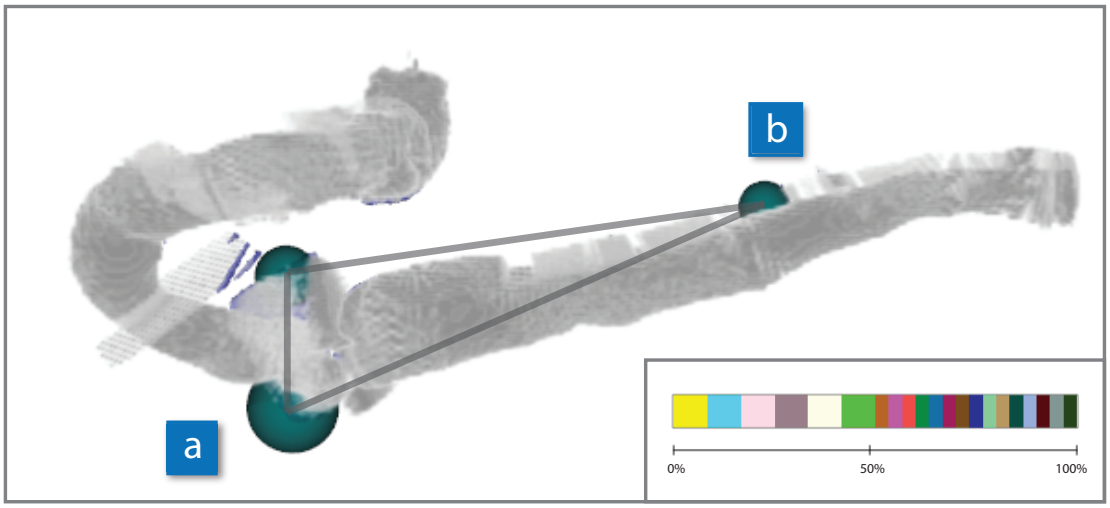

Figure 4.12: Dataset *Aorta*. In the upper part of the aorta (a), six ensemble members could be identified that were responsible for producing artifacts in this region. The corresponding stacked bar-chart can be seen at the lower right. In the lower part of the aorta one region could be identified that contains a dissection, which could not be segmented by all ensemble members (b). The six ensemble members that produce artifacts in (a) are also the only ones that are able to segment the dissection in this region.

in all cases do not exist yet. It is common that segmentations have to be edited afterwards before the analysis can be started. One goal of analyzing the segmentation masks was to find out which mask would require the least manual editing steps afterwards. We therefore concentrated on the high variance regions in the data, and placed probing widgets there. There we could identify six ensemble members where the segmentation of the TL tends to flow into other tissues surrounding the aorta (Figure 4.12a). We then placed a probing widget in another part of the dataset, where, according to what is already known, the aorta is dissected. The variance was high in that region, because not all segmentations were able to capture the two blood flow channels at this position. We then selected the previously identified six ensemble members, and compared them to the rest of the ensemble in the local region. Here it turned out that the six ensemble members were able to capture the two blood flow channels (Figure 4.12b). Our collaboration partners then decided to continue working with the six ensemble members, because it was important that the lower parts of the aorta are segmented properly.

4.5 Summary

In this chapter we presented a visual analysis technique for the exploration of local features in an ensemble of volumetric segmentation masks. We calculate the per-voxel variance and outlier of the ensemble data. The spatial median, the so-called medoid, as evaluated by the Weiszfeld algorithm, is used as a representative of the ensemble data. The regions of high variance are

shown in the same view as the medoid. These regions are rendered as mesh data, to clearly separate them from the medoid data. Local regions in the data can be explored by 3D probing widgets. These widgets can be positioned by mouse interaction, and show the data characteristics at the current position in the separate detail view. The widgets can also be fixed at positions of interest. To visualize similarities between local regions, the probing widgets are arranged in a similarity graph. The probes are represented by nodes, and the edges in the graph indicate whether local regions are similar. The similarity is calculated by analyzing the statistical distribution of the local regions. The edges in the graph are drawn and colored differently, according to whether the whole ensemble should be analyzed, or whether one or more ensemble members should be compared against the rest of the ensemble. With the similarity graph users can interpret local features in a global context.

There are a lot of existing techniques for visualizing **graphs** in 2D, especially if the graphs are very large. The 2D representation could, for example, be replaced by a similarity matrix. Such representations are favorable when searching for patterns in the data, especially if the graph is very large. We, however, consider the 3D graph to be more important. In the 3D graph the spatial position of the nodes is clearly visible. This information is lost in the 2D representation. The main interactions are carried out in the 3D view, and the graph is updated there at every mouse move. This way the users get immediate feedback if they touched a region similar to other nodes in the graph.

Our technique helps users to quickly identify ensemble members that produce different results in local regions of the data. In some cases the **variance** is high, but no local outliers can be detected in the data. This is, for example, the case if segmentations produce slightly different results at the border of an object. In these cases the differences between the ensemble members are more or less uniformly distributed in the stacked bar chart, and no local outliers can be detected (Figure 4.13). On the other hand, such a case also gives some additional information about the data. It shows that in this area all ensemble members are producing slightly different results. Ssuch regions are therefore obviously difficult to segment.

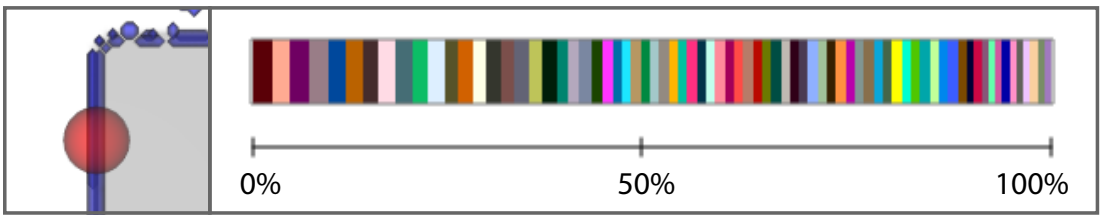

Figure 4.13: Absence of outliers. In the case no outliers can be detected at a local region, the stacked bar chart shows that the differences among the ensemble members are more or less uniformly distributed. Then no outlier can be spotted. The visualization, however, in this case conveys to the user that all ensemble members produce slightly different results in this region.

So far users can define the local regions they are interested in by placing the probing widgets. In the future, we would like to explore means to **automatically extract similar regions** from the data. This is not a trivial task, though. The interesting regions are of unknown size, they can be

smaller/larger than the already defined ones. Therefore, a fixed search kernel would not work. If the kernel is set too large/small, regions might be missed. Also just placing search kernels along fixed grid positions might lead to interesting regions being missed. We would therefore like to implement a flexible technique that is able to detect similar regions. Such a tool would definitely be a great extension for our proposed visualization technique.

Depending on the particular hardware configuration and the size of the ensemble dataset, our difference computations may run into **memory** problems (see Table 4.1 for details). This is due to the fact that it is necessary to calculate a distance map and a surface representation for every input volume, which means that another volume as large as the original one has to be kept in memory. In addition, per-voxel values for the mean, the median and the median absolute deviation have to be stored. New volumes for the variance values and the outliers have to be created. The volume holding the statistical parameters is deleted after the difference computation, though. We therefore implemented a caching mechanism, where only the datasets currently required for the calculation are loaded into the memory. This decreases the memory requirements of the applications, but increases the computation time, since additional reads on the hard-disk become necessary.

The runtime for the **medoid computation** highly depends on the dataset size, and on the number of iterations. Since we operate on a very small sample set compared to the available space (comparing the $width * height * depth$ dimensional space to the n input volumes), the Weiszfeld algorithm converges very quickly. We therefore discovered that approximately 10 iterations are enough to let the algorithm converge in a point.

CHAPTER 5

Conclusion

In this thesis three contributions to the field of comparative visualization of ensembles have been presented. The techniques allow the comparison of ensembles of different data types (in 2D and 3D), and they are scalable to a large number of ensemble members. The thesis is now concluded in this chapter by first summing up the contributions, which were developed in the course of this thesis (Section 5.1). Afterwards, an outlook is given with possibilities for future work (Section 5.2). The chapter is concluded with critical reflections on the topic and the research that has been carried out in the course of this thesis (Section 5.3).

5.1 Contributions

VAICo: Visual Analysis for Image Comparison [SGB13] is a visualization technique for the interactive comparative visualization of image ensembles (Chapter 2). The input images were similar, apart from localized changes, and no reference image was available. We computed the pixel-wise differences in the data and presented them to the user. Interactive visualization tools were provided to explore the image space and drill-down on individual variances. The drill-down allows users to find out what caused the differences in the data. Our visualization approach addresses the scalability of image comparisons and proposes ways to integrate contextual information and more detailed information in one view. Contextual information is preserved, whereas image variances can be efficiently spotted and put into context. Our approach fulfills the visual analytics mantra [Shn96] of *analyze first* (computing the differences), *show the important* (colored polygons), *zoom* (RoD widgets), *filter* (hierarchical clustering) and *analyze further* (interaction possibilities), *details on demand* (access to individual ensemble members). Our approach can be applied to quickly identify small local differences in an image set. It is also helpful for analyzing the occurrence and value ranges of previously defined image features. We demonstrated the scalability and usefulness of our technique by applying it to five image ensembles with varying numbers of images.

YMCA - Your Mesh Comparison Application [SPA+14] described an application suited for the comparison of ensembles of 3D meshes (Chapter 3). A reference mesh was available in every input dataset. Therefore, we could analyzed the data by comparing the input meshes with the ground truth data. We then visually encoded regions of high variance in the data, to highlight regions where the meshes exhibited different properties. The interaction was started with an overview of the data in 3D. Then users could use a 3D magic-lens widget to explore local regions in more detail. The magic-lens widget was linked with a detail view, where more information about the local regions could be shown. Local regions could also be arranged in a parallel coordinates plot. In this plot the local regions were depicted by the axes, and the input meshes were depicted by the polylines. The parallel coordinates plot allowed users to track the error rate of input meshes in different local regions. Our visualization approach combines the concepts of explicit encoding, by showing the high-variance regions in the data, and juxtaposition in the detail view, where different input meshes could be viewed side-by-side. The proposed system fulfills the visual analytics mantra of *analyze first* (computing the differences), *show the important* (colored variance regions), *zoom* (magic-lens widgets), *filter* (parallel coordinates) and *analyze further* (interaction possibilities), *details on demand* (access to individual ensemble members). YMCA further addresses the scalability problem of previous mesh comparison tools. We tested our system with mesh datasets from point cloud reconstruction, and YMCA turned out to be useful to identify outliers and patterns in the data.

Visual Analysis of Volume Ensembles Based on Local Features [SFP+16] aimed at the analysis of local regions in a volume ensemble dataset (Chapter 4). The input data consisted of a set of volume dataset, without having ground truth data available. We calculated the per-voxel variance and outlier of the ensemble data. The spatial median, the so-called medoid, as evaluated by the Weiszfeld algorithm, was used as a representative of the ensemble data. The regions of high variance were shown in the same view as the medoid. Local regions in the data could be explored by 3D probing widgets. These widgets could be positioned by mouse interaction, and showed the data characteristics at the current position in the separate detail view. To visualize similarities between local regions, the probing widgets were arranged in a similarity graph. The probes were depicted by the nodes, and the edges in the graph indicated whether local regions were similar (i.e., exhibiting an equal statistical distribution). The edges in the graph were colored differently, according to whether the whole dataset was analyzed, or whether one or more ensemble members were compared against the rest of the ensemble. The visual analytics mantra of *analyze first* (computing the variance), *show the important* (rendering of the medoid and the high-variance regions), *zoom* (probing widgets), *filter* (similarity graph) and *analyze further* (stacked-bar charts), *details on demand* (access to individual ensemble members) is fulfilled within this approach. As an exemplary use case, we applied our technique to ensembles of volumetric segmentation masks. The masks were created by testing segmentation algorithms with different parameter settings. Our technique proved to be useful to link similar local regions, to compare ensemble members against the rest of the ensemble, and to interpret local features in a global context.

5.2 Outlook

Comparative visualization and ensemble visualization are emerging topics in visualization. Due to the availability of ensemble datasets in various different domains, findings from these fields will become interesting and useful for several application areas. The following research directions may be considered in the future.

Localized Differences The first two techniques presented in this thesis (VAICo [SGB13] and YMCA [SPA+14]) have been designed for input datasets that are mostly similar, but exhibit small localized changes. Although a lot of datasets exist that fulfill this requirement, as outlined in the papers, an interesting challenge would be to further explore the visualization of global changes. VAICo, for instance, would fail if applied to images that are very dissimilar in pixel color (e.g., landscape images). It would be not possible, for example, to use VAICo to track changes in yearly pictures of glaciers or surveillance video frames. Similarly, YMCA would fail if input meshes greatly differ from the given reference mesh (e.g., if mesh parts are missing). It would therefore be interesting to explore visualization techniques that can display global changes in a dataset. Another challenge is to track changes that greatly change their shape (e.g., grow/shrink).

Comparison Metrics Similar to the question how visualizations can deal with global changes, an interesting research direction is to develop appropriate comparison metrics for ensemble data. For the techniques presented in this thesis, rather simple comparison metrics have been used. This was due to the fact that the main focus of our work was on the visualization of the differences in the data, and on the interaction possibilities. However, more advanced metrics would be favorable for the comparison of ensemble datasets. In the case of 2D images, several other metrics have already been proposed [ZCW02] which could be applied. For 3D data, especially volume data, the development of appropriate comparison metrics is a challenging problem. Some approaches use derived features like iso-surfaces [FML16] to compare the data. The problem with finding a proper metric is, that it in many cases highly depends on the task and on the available data (e.g., for comparing molecular surfaces [SKR+14]).

Find and Compare Features We also came across the question of how to automatically extract interesting regions from the data. In YMCA [SPA+14], we used a Mean-Shift based approach to detect the most interesting high-variance regions. This worked nicely, because we only had to consider positions on the mesh surface. In our third approach for comparing volumes [SFP+16], we let users decide in which local regions they are interested in. The user-defined positions were then connected in a graph. As a future work, it would be interesting to develop techniques that could automatically detect similar regions, based on the initial user selection. The idea would be that users define some regions, and the algorithm automatically detects similar ones. This is not a trivial task, though. The interesting regions are of unknown size, they can be smaller/larger than the already defined ones. Therefore, a fixed search kernel would not work. If the kernel is set to large/small, regions might be missed. Also just placing search kernels along fixed grid positions might lead to interesting regions being missed. Obermaier and Joy [OJ15] recently proposed a technique for the automatic placement of slice planes in a volume dataset. Based on this line of

thinking, a search function could be implemented for the automatic search for similar regions. Such a technique would definitely be a great extension for our proposed visualization technique for volume dataset ensembles, as it can then be assured that important regions are not missed during the analysis.

Summarization In this thesis we concentrated on the visual analysis of differences in ensemble datasets. According to the findings of the local analysis, users can draw conclusions from the data. A different research direction would be to find ways to provide a concise overview of the data to the user. Such an overview should convey the variability of the dataset, probably by dividing the data into groups or clusters. Further, outlier should be clearly visible in the overview. The overview should also fit into one view, so that it can be easily grasped by the users. In the case of images, approaches have already been presented to automatically create picture collages [TSLX12]. Such techniques, however, rather focus on aesthetics than on the analysis aspect of the representation. A concise overview of an image ensemble could be, for example, very useful to evaluate images from surveillance cameras. For 3D ensembles, like mesh or volume data, no techniques are currently available that would create a concise overview of the data.

Representations and Interaction The representation of differences and the interaction with the multitude of information available in ensembles is a challenging task. It is especially challenging, if the data should be represented in 3D. Visualizations in this case have to deal with occlusions and perspective distortions, and all user interactions also have to consider the third dimension. 3D representations are also especially challenging for comparative visualization and ensemble visualization. There are many open problems for which no suitable visualization technique exists yet. For example, iso-surfaces are an important concept for the analysis of weather data ensembles. Iso-surfaces can be used to encode temperature differences, or flow like wind. However, no appropriate visualization technique exists yet that allows to display several iso-surfaces in 3D in a way that differences between them are also visible. There are already some approaches trying to tackle the problem, for example by Demir et al. [DKW16]. They decided to use silhouettes, so that the displayed information is reduced, although the major shape of the surfaces is still visible. This topic definitely provides some interesting possibilities for future research works. A similar problem exists for iso-contours in 2D and 3D, due to the fact that spaghetti plots very easily suffer from over-plotting. The techniques for ensemble visualization mentioned in Section 1.2.2 mainly try to solve this problem by applying statistical measurements. This, however, hides the underlying raw data, and in this way might hinder the proper understanding of the visualization. New techniques for *unscrambling* spaghetti plots in 2D and 3D are definitely an interesting topic for future research.

5.3 Reflections

After finishing this thesis, I now critically reflect on the topic and the four years of research. When I started this thesis, the topic of comparative visualization was already an emerging topic in visualization. The term ensemble was at that time only used for simulation data. We at that time decided to extend existing comparative visualization techniques to make them scalable to a large number of datasets. This topic has now become even more interesting, since more and more ensemble datasets are created now in different application areas. Ensemble visualization nowadays should therefore not any longer be only limited to simulation data.

A very challenging problem in the future will be the amount of data that has to be processed. Techniques will need to be efficient enough to handle a large amount of data in reasonable time. Memory on the graphics card, though, is limited, and it will not be possible to load whole ensembles onto the graphics card to process them.

When defining new metrics for comparison methods, a big challenge is to consider the semantics. The more sophisticated a comparison technique will be, the more targeted it will be towards a certain task.

I still think that the topic of summarization would be a very interesting one for the future. There are cases where users do not want to spend time analyzing the dataset in detail. Sometimes they are fine with getting a quick overview, so that they can concentrate on the outliers in the data. In other cases it would be important to quickly identify patterns or groups in the data. Therefore, I believe that visualization techniques for providing a quick, concise overview of an ensemble would be a great help for many application areas.

List of Figures

List of Tables

Bibliography

[AH11] Paolo Angelelli and Helwig Hauser. Straightening Tubular Flow for Side-by-Side Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2063–2070, Dec. 2011.

[AK04] Nina Amenta and Yong Joo Kil. Defining Point-Set Surfaces. *ACM Transactions on Graphics*, 23(3):264–270, Aug. 2004.

[AMST11] Wolfgang Aigner, Silvia Miksch, Heidrun Schuman, and Christian Tominski. *Visualization of Time-Oriented Data*. Human-Computer Interaction. Springer Verlag, London, UK, 1st edition, 2011.

[ASCE02] Nicolas Aspert, Diego Santa-Cruz, and Touradji Ebrahimi. MESH: Measuring Error between Surfaces using the Hausdorff Distance. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, ICME '02, pages 705–708. IEEE, Aug. 2002.

[AWH+12] Oluwafemi S. Alabi, Xunlei Wu, Jonathan M. Harter, Madhura Phadke, Lifford Pinto, Hannah Petersen, Steffen Bass, Michael Keifer, Sharon Zhong, Christopher G. Healey, and Russell M. Taylor. Comparative visualization of ensembles using ensemble surface slicing. In *Proceedings of the SPIE, Visualization and Data Analysis*, volume 8294, 2012.

[AWW09] Keith Andrews, Martin Wohlfahrt, and Gerhard Wurzinger. Visual Graph Comparison. In *Proceedings of the 13th International Conference on Information Visualisation*, pages 62–67, July 2009.

[BBF+11] Stef Busking, Charl P. Botha, Luca Ferrarini, Julien Milles, and Frits H. Post. Image-based rendering of intersecting surfaces for dynamic comparative visualization. *Vis Comput*, 27(5):347–363, 2011.

[BC87] Richard A. Becker and William S. Cleveland. Brushing Scatterplots. *Technometrics*, 29(2):127–142, May 1987.

[BHE+15] Dorit Borrmann, Robin Hess, Daniel Eck, Hamidreza Houshiar, Andreas Nüchter, and Klaus Schilling. Evaluation of Methods for Robotic Mapping of Cultural Heritage Sites. In *Proceedings of the 2nd IFAC Conference on Embedded Systems, Computer Intelligence and Telematics*, CESCIT '15, pages 105–110, June 2015.

[BHGK14] Michael Beham, Wolfgang Herzner, M. Eduard Gröller, and Johannes Kehrer. Cupid: Cluster-Based Exploration of Geometry Generators with Parallel Coordinates and Radial Trees. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1693–1702, Dec 2014.

[BLN+13] Matthew Berger, Joshua A. Levine, Luis Gustavo Nonato, Gabriel Taubin, and Claudio T. Silva. A Benchmark for Surface Reconstruction. *ACM Transactions on Graphics*, 32(2):20:1–20:17, Apr. 2013.

[BPR+15] Michael Böttinger, Holger Pohlmann, Niklas Röber, Karin Meier-Fleischer, and Dela Spickermann. Visualization of 2D Uncertainty in Decadal Climate Predictions. In *Workshop on Visualisation in Environmental Sciences*, EnvirVis '15. The Eurographics Association, 2015.

[BS14] Amir Beck and Shoham Sabach. Weiszfeld's Method: Old and New Results. *Journal of Optimization Theory and Applications*, 164(1):1–40, May 2014.

[BSM+13] Steven Bergner, Michael Sedlmair, Torsten Möller, Sareh N. Abdolyousefi, and Ahmed Saad. ParaGlide: Interactive Parameter Space Partitioning for Computer Simulations. *IEEE Transactions on Visualization and Computer Graphics*, 19(9):1499–1512, Sept. 2013.

[BTS+14] Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Joshua A. Levine, Andrei Sharf, and Claudio T. Silva. State of the Art in Surface Reconstruction from Point Clouds. In Sylvain Lefebvre and Michela Spagnuolo, editors, *Proceedings of the Eurographics 2014, Eurographics STARs*, EG '14, pages 161–185. Eurographics Association, 2014.

[BVB+11] Michael Burch, Corinna Vehlow, Fabian Beck, Stephan Diehl, and Daniel Weiskopf. Parallel Edge Splatting for Scalable Dynamic Graph Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2344–2353, Dec. 2011.

[CBC+01] Jonathan C. Carr, Richard K. Beatson, Jon B. Cherrie, Tim J. Mitchell, W. Richard Fright, Bruce C. McCallum, and Tim R. Evans. Reconstruction and Representation of 3D Objects with Radial Basis Functions. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 67–76, New York, NY, USA, 2001. ACM.

[CCR08] Paolo Cignoni, Massimiliano Corsini, and Guido Ranzuglia. MeshLab: an Open-Source 3D Mesh Processing System. *ERCIM News*, 2008(73), 2008.

[CKB09] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys*, 41(1):2:1–2:31, Jan 2009.

[CLEK13] Dane Coffey, Chi-Lun Lin, Arthur G. Erdman, and Daniel F. Keefe. Design by Dragging: An Interface for Creative Forward and Inverse Design with Simulation Ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2783–2791, Dec. 2013.

[CM02] Dorin Comaniciu and Peter Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Transations on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

[CP99] Brian Clarkson and Alex Pentland. Unsupervised clustering of ambulatory audio and video. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, ICASSP '99, pages 3037–3040, Washington, DC, USA, Mar. 1999. IEEE Computer Society.

[CRS98] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. Metro: Measuring Error on Simplified Surfaces. *Compututer Graphics Forum*, 17(2):167–174, 1998.

[CV01] Tony Chan and Luminita Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.

[DDW14] Ismail Demir, Christian Dick, and Rüdiger Westermann. Multi-Charts for Comparative 3D Ensemble Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 20(12), Dec. 2014.

[DH73] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons Inc, 1973.

[DH02] Helmut Doleisch and Helwig Hauser. Smooth Brushing for Focus+Context Visualization of Simulation Data in 3D. In *Proceedings of the Winter School of Computer Graphics*, WSCG '02, pages 147–154, Plzen-Bory, Czech Republic, Feb. 2002.

[DKW16] Ismail Demir, Johannes Kehrer, and Rüdiger Westermann. Screen-space silhouettes for visualizing ensembles of 3d isosurfaces. In *Proceedings of IEEE PacificVis Visualization Notes*, PACIFICVIS '16, Apr. 2016. accepted for publication.

[dLvW93] Willem C. de Leeuw and Jarke J. van Wijk. A Probe for Local Flow Field Visualization. In *Proceedings of the 4th Conference on Visualization*, VIS '93, pages 39–45, Washington, DC, USA, 1993. IEEE Computer Society.

[DPA06] Steven M. Drucker, Georg Petschnigg, and Maneesh Agrawala. Comparing and managing multiple versions of slide presentations. In *Proceedings of the 19th Annual ACM symposium on User Interface Software and Technology*, UIST '06, pages 47–56, New York, NY, USA, Oct. 2006. ACM.

[DPD+15] Alexandra Diehl, Leandro Pelorosso, Claudio Delrieux, Celeste Saulo, Juan Ruiz, M. Eduard Gröller, and Stefan Bruckner. Visual Analysis of Spatio-Temporal Data:

Applications in Weather Forecasting. *Compututer Graphics Forum*, 34(3):381–390, 2015.

[DZDL02] Marco DaSilva, Song Zhang, Cagatay Demiralp, and David H. Laidlaw. Visualizing the Differences between Diffusion Tensor Volume Images. In *Proceedings of the ISMRM Workshop on Diffusion MRI: What Can We Measure?*, ISMRM '02, pages 237–238, St. Malo, France, Mar. 2002.

[EBR07] Étienne Baudrier and Alain Riffaud. A Method for Image Local-Difference Visualization. In *Proceedings of the 9th International Conference on Document Analysis and Recognition*, volume 2 of *ICDAR '07*, pages 949–953, Sept. 2007.

[EF08] Niklas Elmqvist and Jean-Daniel Fekete. Semantic Pointing for Object Picking in Complex 3D Environments. In *Proceedings of Graphics Interface*, GI '08, pages 243–250, Toronto, Ont., Canada, 2008. Canadian Information Processing Society.

[EKLN04] Cesim Erten, Stephen G. Kobourov, Vu Le, and Armand Navabi. Simultaneous Graph Drawing: Layout Algorithms and Visualization Schemes. In *Proceedings of the 11th International Symposium on Graph Drawing*, GD '03, pages 437–449, Berlin, Heidelberg, Germany, Sept. 2004. Springer Berlin Heidelberg.

[ENP+08] Danilo M. Eler, Marcel Y. Nakazaki, Fernando V. Paulovich, Davi P. Santos, M. Cristina F. Oliveira, Jo ao E. S. Batisto Neto, and Rosane Minghim. Multidimensional Visualization to Support Analysis of Image Collections. In *Proceedings of the XXI Brazilian Symposium on Computer Graphics and Image Processing*, SIBGRAPI '08, pages 289–296, Washington, DC, USA, 2008. IEEE Computer Society.

[ESBB98] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, Dec. 1998.

[EST07] Niklas Elmqvist, John Stasko, and Philippas Tsigas. DataMeadow: A Visual Canvas for Analysis of Large-Scale Multivariate Data. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, VAST '07, pages 187–194, Oct. 2007.

[FBW16] Florian Ferstl, Kai Bürger, and Rüdiger Westermann. Streamline Variability Plots for Characterizing the Uncertainty in Vector Field Ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):767–776, Jan 2016.

[FH09] Raphael Fuchs and Helwig Hauser. Visualization of Multi-Variate Scientific Data. *Computer Graphics Forum*, 28(6):1670–1690, 2009.

[FHK+08] Charless C. Fowlkes, Cris L. Luengo Hendriks, Soile V. E. Kerănen, Gunther H. Weber, Oliver Rübel, Min-Yu Huang, Sohail Chatoor, Angela H. DePace, Lisa Simirenko, Clara Henriquez, Amy Beaton, Richard Weiszmann, Susan Celniker,

Bernd Hamann, David W. Knowles, Mark D. Biggin, Michael B. Eisen, and Jitendra Malik. A Quantitative Spatiotemporal Atlas of Gene Expression in the Drosohpila Blastoderm. *Cell*, 133:364–374, 2008.

[FHM16] Bernhard Fröhler, Christoph Heinzl, and Torsten Möller. GEMSe: Visualization-Guided Exploration of Multi-channel Segmentation Algorithms. *Computer Graphics Forum*, 35(3), 2016. accepted for publication.

[FML16] Alexey Fofonov, Vladimir Molchanov, and Lars Linsen. Visual Analysis of Multi-run Spatio-temporal Simulations Using Isocontour Similarity for Projected Views. *IEEE Transactions on Visualization and Computer Graphics*, 2016. accepted for publication.

[GAW+11] Michael Gleicher, Danielle Albers, Rick Walker, Ilir Jusufi, Charles D. Hansen, and Jonathan C. Roberts. Visual Comparison for Information Visualization. *Information Visualization*, 10(4):289–309, Oct 2011.

[GBP+13] Luke Gosink, Kevin Bensema, Trenton Pulsipher, Harald Obermaier, Michael Henry, Hank Childs, and Kenneth I. Joy. Characterizing and Visualizing Predictive Uncertainty in Numerical Ensembles Through Bayesian Model Averaging. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2703–2712, Dec. 2013.

[GJP+14] Marc G. Genton, Christopher Johnson, Kristin Potter, Georgiy Stenchikov, and Ying Sun. Surface boxplots. *Stat*, 3(1):1–11, 2014.

[Gra05] Leo Grady. Multilabel Random Walker Image Segmentation Using Prior Models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 763–770, June 2005.

[GSK+15] Alexander Geurts, Georgios Sakas, Arjan Kuijper, Meike Becker, and Tatiana von Landesberger. Visual Comparison of 3D Medical Image Segmentation Algorithms Based on Statistical Shape Models. In *Lecture Notes in Computer Science*, pages 336–344. Springer, 2015.

[HCJ+14] Charles D. Hansen, Min Chen, Christopher R. Johnson, Arie E. Kaufman, and Hans Hagen, editors. *Scientific Visualization*. Springer-Verlag London, 2014.

[HDD+92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface Reconstruction from Unorganized Points. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, pages 71–78, New York, NY, USA, 1992. ACM.

[Hin09] Hans Hinterberger. *Encyclopedia of Database Systems*, chapter Comparative Visualization, pages 405–405. Springer US, Boston, MA, USA, 2009.

[HMZ+14] Thomas Höllt, Ahmed Magdy, Peng Zhan, Guoning Chen, Ganesh Gopalakrishnan, Ibrahim Hoteit, Charles D. Hansen, and Markus Hadwiger. Ovis: A Framework for Visual Analysis of Ocean Forecast Ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1114–1126, Aug. 2014.

[HOGJ13] Mathias Hummel, Harald Obermaier, Christoph Garth, and Kenneth I. Joy. Comparative Visual Analysis of Lagrangian Transport in CFD Ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2743–2752, Dec. 2013.

[How14] David C. Howell. *Wiley StatsRef: Statistics Reference Online*, chapter Median Absolute Deviation. John Wiley & Sons, Ltd, 2014.

[HRTV06] Benjamin V. Hollingsworth, Stephan E. Reichenbach, Qingping Tao, and Arvind Visvanathan. Comparative visualization for comprehensive two-dimensional gas chromatography. *Journal of Chromatography A*, 1105(1-2):51–8, 2006.

[HSKIH06] Haleh Hagh-Shenas, Sunghee Kim, Victoria Interrante, and Christopher Healey. Weaving versus blending: a quantitative assessment of the information carrying capacities of two alternative methods for conveying multivariate data with color. In *Proceedings of the 3rd Symposium on Applied Perception in Graphics and Visualization*, APGV '06, pages 164–164, New York, NY, USA, July 2006. ACM.

[HSNHH10] Ingrid Hotz, Jaya Sreevalsan-Nair, Hans Hagen, and Bernd Hamann. Tensor Field Reconstruction Based on Eigenvector and Eigenvalue Interpolation. In Hans Hagen, editor, *Scientific Visualization: Advanced Concepts*, volume 1 of *Dagstuhl Follow-Ups*, pages 110–123. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2010.

[HSSK16] Max Hermann, Anja C. Schunke, Thomas Schultz, and Reinhard Klein. Accurate Interactive Visualization of Large Deformations and Variability in Biomedical Image Ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):708–717, Jan. 2016.

[ID90] Alfred Inselberg and Bernard Dimsdale. Parallel Coordinates: A Tool for Visualizing Multi-dimensional Geometry. In *Proceedings of the 1st Conference on Visualization*, VIS '90, pages 361–370, Washington, DC, USA, 1990. IEEE Computer Society.

[IEGC08] Tobias Isenberg, Maarten H. Everts, Jens Grubert, and Sheelagh Carpendale. Interactive Exploratory Visualization of 2D Vector Fields. *Computer Graphics Forum*, 27(3):983–990, 2008.

[IH93] Boris Iglewicz and David Hoaglin. *How to Detect and Handle Outliers*, volume 16. ASQC/Quality Press, 1993.

[Jac16] Mathieu Jacomy. i want hue - Colors for data scientists. http://tools.medialab.sciences-po.fr/iwanthue, 2016. [Online; accessed 24-March-2016].

[JDKW15] Mihaela Jarema, Ismail Demir, Johannes Kehrer, and Rüdiger Westermann. Comparative visual analysis of vector field ensembles. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, VAST '15, pages 81–88, Oct. 2015.

[JSW05] Tao Ju, Scott Schaefer, and Joe Warren. Mean Value Coordinates for Closed Triangular Meshes. *ACM Transactions on Graphics*, 24(3):561–566, July 2005.

[JYC+10] Radu Jianu, Kebing Yu, Lulu Cao, Vinh Nguyen, Arthur R. Salomon, and David H. Laidlaw. Visual integration of quantitative proteomic data, pathways and protein interactions. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):609–620, 2010.

[KAC+12] Amin Katouzian, Elsa D. Angelini, Stéphane G. Carlier, Jasjit S. Suri, Nassir Navab, and Andrew F. Laine. A State-of-the-Art Review on Segmentation Algorithms in Intravascular Ultrasound (IVUS) Images. *IEEE Transactions on Information Technology in Biomedicine*, 16(5):823–834, Sept. 2012.

[KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 61–70. Eurographics Association, 2006.

[KBKG09] Peter Kohlmann, Stefan Bruckner, Armin Kanitsar, and M. Eduard Gröller. Contextual picking of volumetric structures. In *Proceedings of the IEEE Pacific Visualization Symposium*, PacificVis '09, pages 185–192, Apr. 2009.

[KERC09] Daniel Keefe, Marcus Ewert, William Ribarsky, and Remco Chang. Interactive Coordinated Multiple-View Visualization of Biomechanical Motion Data. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1383–1390, Nov. 2009.

[KH13] Johannes Kehrer and Helwig Hauser. Visualization and Visual Analysis of Multifaceted Scientific Data: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):495–513, March 2013.

[KHZ04] Paul Kammerer, Alan Hanbury, and Ernestine Zolda. A Visualisation Tool for Comparing Paintings and their Underdrawings. In *Proceedings of the Conference on Electronic Imaging for the Visual & Performing Arts*, EVA '04, pages 148–153, July 26-31 2004.

[KKH02] Joe Kniss, Gordon Kindlmann, and Charles D. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, Jul. 2002.

[KMDH11] Johannes Kehrer, Philipp Muigg, Helmut Doleisch, and Helwig Hauser. Interactive Visual Analysis of Heterogeneous Scientific Data across an Interface. *IEEE Transactions on Visualization and Computer Graphics*, 17(7):934–946, July 2011.

[KPBG13] Johannes Kehrer, Harald Piringer, Wolfgang Berger, and M. Eduard Gröller. A Model for Structure-Based Comparison of Many Categories in Small-Multiple Displays. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2287–2296, Dec. 2013.

[LC87] William E. Lorensen and Harvey E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *ACM Transactions on Graphics*, 21(4):163–169, Aug. 1987.

[LKH10] Ove Daae Lampe, Johannes Kehrer, and Helwig Hauser. Visual Analysis of Multivariate Movement Data Using Interactive Difference Views. In *Proceedings of Vision, Modeling, and Visualization*, VMV '10, pages 315–322, 2010.

[LPK05] Alison L. Love, Alex Pang, and David L. Kao. Visualizing spatial multivalue data. *IEEE Computer Graphics and Applications*, 25(3):69–79, May 2005.

[LSP+10] Alexander Lex, Marc Streit, Christian Partl, Karl Kashofer, and Dieter Schmalstieg. Comparative Analysis of Multidimensional, Quantitative Data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1027–1035, 2010.

[LWZ+16] Zhaoxin Li, Kuanquan Wang, Wangmeng Zuo, Deyu Meng, and Lei Zhang. Detail-Preserving and Content-Aware Variational Multi-View Stereo Reconstruction. *IEEE Transactions on Image Processing*, 25(2):864–877, Feb 2016.

[MBS+12] Gabriel Mistelbauer, Hamed Bouzari, Rüdiger Schernthaner, Ivan Baclija, Arnold Köchl, Stefan Bruckner, Milos Šramek, and M. Eduard Gröller. Smart Super Views – A Knowledge-assisted Interface for Medical Visualization. In *Proceedings of the 2012 IEEE Conference on Visual Analytics Science and Technology*, VAST '12, pages 163–172, Washington, DC, USA, 2012. IEEE Computer Society.

[MEV+05] Matej Mlejnek, Pierre Ermes, Anna Vilanova, Rob van der Rijt, Harrie van den Bosch, Frans Gerritsen, and M. Eduard Gröller. Profile Flags: A Novel Metaphor for Probing of T2 Maps. In *Proceedings of the IEEE Conference on Visualization*, VIS '05, page 76, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[MGJ+10] Krešimir Matković, Denis Gračanin, Mario Jelović, Andreas Ammer, Alan Lež, and Helwig Hauser. Interactive Visual Analysis of Multiple Simulation Runs Using the Simulation Model View: Understanding and Tuning of an Electronic Unit Injector. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1449–1457, 2010.

[MGKH09] Krešimir Matković, Denis Gračanin, Borislav Klarin, and Helwig Hauser. Interactive Visual Analysis of Complex Scientific Data as Families of Data Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1351–1358, Nov. 2009.

[MGT+03] Tamara Munzner, François Guimbretiére, Serdar Tasiran, Li Zhang, and Yunhong Zhou. TreeJuxtaposer: Scalable Tree Comparison Using Focus+Context with Guaranteed Visibility. *ACM Transactions on Graphics*, 22(3):453–462, July 2003.

[MHG10] Muddassir M. Malik, Christoph Heinzl, and M. Eduard Gröller. Comparative Visualization for Parameter Studies of Dataset Series. *IEEE Transactions on Visualization and Computer Graphics*, 16(5):829–840, Sep. 2010.

[MIA+03] Tomohito Masuda, Setsuo Imazu, Supatana Auethavekiat, Tsuyoshi Furuya, Kunihiko Kawakami, and Katsushi Ikeuchi. Shape difference Visualization for ancient bronze Mirrors through 3D range images. *Journal of Visualization and Computer Animation*, 14(4):183–196, 2003.

[Mil07] James R. Miller. Attribute Blocks: Visualizing Multiple Continuously Defined Attributes. *IEEE Computer Graphics and Applications*, 27:57–69, May 2007.

[MWK14] Mahsa Mirzargar, Ross T. Whitaker, and Robert M. Kirby. Curve Boxplot: Generalization of Boxplot for Ensembles of Curves. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2654–2663, Dec. 2014.

[(NC14] NCAR Command Language (NCL). NCL Data Analysis and Visualization Software, Climate Variability Diagnostics Package (CVDP). https://nar.ucar.edu/sites/default/files/labs/cisl/2014/1404-450.png, 2014. [Online; accessed 07-April-2016].

[NFB07] Thomas Nocke, Michael Flechsig, and Uwe Böhm. Visual exploration and evaluation of climate-related simulation data. In *Proceedings of the Simulation Conference, Winter 2007*, pages 703–711, Dec. 2007.

[NSGS07] Galileo M. Namata, Brian Staats, Lise Getoor, and Ben Shneiderman. A Dual-view Approach to Interactive Network Visualization. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management*, CIKM '07, pages 939–942, New York, NY, USA, 2007. ACM.

[OJ14] Harald Obermaier and Kenneth I. Joy. Future challenges for ensemble visualization. *IEEE Computer Graphics and Applications*, 34(3):8–11, May 2014.

[OJ15] H. Obermaier and K. I. Joy. An Automated Approach for Slicing Plane Placement in Visual Data Analysis. *IEEE Transactions on Visualization and Computer Graphics*, 21(12):1403–1414, Dec. 2015.

[PBVG10] Daniel Patel, Stefan Bruckner, Ivan Viola, and M. Eduard Gröller. Seismic volume visualization for horizon extraction. In *Proceedings of the IEEE Pacific Visualization Symposium*, PacificVis '10, pages 73–80, Mar. 2010.

[PF96] Alex Pang and Adam Freeman. Methods for comparing 3d surface attributes. In *Proceedings of the SPIE, Visualization and Data Analysis*, volume 2656, pages 58–64, 1996.

[PK06] Ted Pedersen and Anagha Kulkarni. Automatic cluster stopping with criterion functions and the gap statistic. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, NAACL-Demonstrations '06, pages 276–279. Association for Computational Linguistics, 2006.

[PPBT12] Harald Piringer, Stephan Pajer, Wolfgang Berger, and Heike Teichmann. Comparative Visual Analysis of 2D Function Ensembles. *Computer Graphics Forum*, 31(3.3):1195–1204, 2012.

[PRW11] Tobias Pfaffelmoser, Matthias Reitinger, and Rüdiger Westermann. Visualizing the Positional and Geometrical Variability of Isosurfaces in Uncertain Scalar Fields. *Computer Graphics Forum*, 30(3):951–960, 2011.

[PWB+09] Kristin Potter, Andrew Wilson, Peer-Timo Bremer, Dean Williams, Charles Doutriaux, Valerio Pascucci, and Chris R. Johnson. Ensemble-Vis: A Framework for the Statistical Visualization of Ensemble Data. In *Proceedings of the IEEE International Conference on Data Mining, Workshops*, ICDMW '09, pages 233–240, Dec. 2009.

[RFT04] Michaël Roy, Sebti Foufou, and Frédéric Truchetet. Mesh Comparison using Attribute Deviation Metric. *International Journal of Image and Graphics*, 4(1):127–140, Jan. 2004.

[RGSW15] Marc Rautenhaus, Christian M. Grams, Andreas Schäfler, and Rüdiger Westermann. Three-dimensional visualization of ensemble weather forecasts - Part 2: Forecasting warm conveyor belt situations for aircraft-based field campaigns. *Geoscientific Model Development*, 8(7):2355–2377, 2015.

[RKSW15] Marc Rautenhaus, Michael Kern, Andreas Schäfler, and Rüdiger Westermann. Three-dimensional visualization of ensemble weather forecasts - Part 1: The visualization tool Met.3D (version 1.0). *Geoscientific Model Development*, 8(7):2329–2353, 2015.

[RMK+15] Mukund Raj, Mahsa Mirzargar, Robert M. Kirby, Ross T. Whitaker, and J. Samuel Preston. Evaluating Alignment of Shapes by Ensemble Visualization. *IEEE Computer Graphics and Applications*, 1(1), 2015.

[RVB+09] Timo Ropinski, Ivan Viola, Martin Biermann, Helwig Hauser, and Klaus Hinrichs. Multimodal Visualization with Interactive Closeups. In *EGUK Theory and Practice of Computer Graphics*, pages 17–24, June 2009.

[SFP+16] Johanna Schmidt, Bernhard Fröhler, Reinhold Preiner, Johannes Kehrer, M. Eduard Gröller, Stefan Bruckner, and Christoph Heinzl. Visual Analysis of Volume Ensembles Based on Local Features. Technical Report TR-186-2-16-2, Institute of Computergraphics and Algorithms, TU Wien, Austria, 2016.

[SGB13] Johanna Schmidt, M. Eduard Gröller, and Stefan Bruckner. VAICo: Visual Analysis for Image Comparison. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2090–2099, Dec. 2013.

[SHB98] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Chapman & Hall, 2 edition, 1998.

[Shn96] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. *IEEE Symposium on Visual Languages*, pages 336—-343, 1996.

[SHP+14] Michael Sedlmair, Christoph Heinzl, Harald Piringer, Stefan Bruckner, and Torsten Möller. Visual Parameter Space Analysis: A Conceptual Framework. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2161–2170, 2014.

[SK07] Dominik Sibbing and Leif Kobbelt. Fast Interactive Region of Interest Selection for Volume Visualization. In *Bildverarbeitung für die Medizin 2007*, volume 12, pages 338–342. Springer Berlin Heidelberg, 2007.

[SKR+14] Katrin Scharnowski, Michael Krone, Guido Reina, Tobias Kulschewski, Jürgen Pleiss, and Thomas Ertl. Comparative Visualization of Molecular Surfaces Using Deformable Models. *Computer Graphics Forum*, 33(3):191–200, 2014.

[SMS05] Samuel Silva, Joaquim Madeira, and Beatriz Sousa Santos. PolyMeCo - A Polygonal Mesh Comparison Tool. In *Proceedings of the 9th International Conference on Information Visualisation*, IV '05, pages 842–847, Washington, DC, USA, Jul. 2005. IEEE Computer Society.

[SPA+14] Johanna Schmidt, Reinhold Preiner, Thomas Auzinger, Michael Wimmer, M. Eduard Gröller, and Stefan Bruckner. YMCA - Your Mesh Comparison Application. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, VAST '14, pages 153–162, Washington, DC, USA, Nov. 2014. IEEE Computer Society.

[ST94] Toyofum Saito and Jun-Ichiro Toriwaki. New algorithms for Euclidean distance transformation of an n-dimensional digitized picture with applications. *Pattern Recognition*, 27(11):1551–1565, 1994.

[Str06] Matús Straka. *Processing and Visualization of Peripheral CT-Angiography Datasets*. PhD thesis, TU Wien, Institute of Computer Graphics and Algorithms, Vienna, Aug. 2006.

[STS06] Natascha Sauber, Holger Theisel, and Hans-Peter Seidel. Multifield-Graphs: An Approach to Visualizing Correlations in Multifield Scalar Data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):917–924, Sept 2006.

[SWMJ99] Nivedita Sahasrabudhe, John E. West, Raghu Machiraju, and Mark Janus. Structured spatial domain image and data comparison metrics. In *Proceedings of the Conference*

on Visualization, VIS '99, pages 97–104, Washington, DC, USA, Oct. 1999. IEEE Computer Society.

[SZ11] Jeff Simmons and Ji-Cheng Zhao. Large dataset generation, integration and simulation in materials science. *Journal of The Minerals, Metals & Materials Society (JOM)*, 63(7):40, 2011.

[SZD+10] Jibonananda Sanyal, Song Zhang, Jamie Dyer, Andrew Mercer, Philip Amburn, and Robert Moorhead. Noodles: A Tool for Visualization of Numerical Weather Model Ensemble Uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1421–1430, Nov. 2010.

[TC05] James J. Thomas and Kristin A. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Center, 2005.

[TFJ12] Christian Tominski, Camilla Forsell, and Jimmy Johansson. Interaction Support for Visual Comparison Inspired by Natural Behavior. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2719–2728, Dec. 2012.

[TSFH+13] Melanie Tory, Sheryl Staub-French, Dandan Huang, Yu-Ling Chang, Colin Swindells, and Rachel Pottinger. Comparative visualization of construction schedules. *Automation in Construction*, 29:68–82, 2013.

[TSLX12] L. Tan, Y. Song, S. Liu, and L. Xie. ImageHive: Interactive Content-Aware Image Summarization. *IEEE Computer Graphics and Applications*, 32(1):46–55, Jan. 2012.

[Tuf86] Edward R. Tufte. *The visual display of quantitative information*. Graphics Press, Cheshire, CT, USA, 1986.

[TWSM+11] Thomas Torsney-Weir, Ahmed Saad, Torsten Möller, Britta Weber, Hans-Christian Hege, Jean-Marc Verbavatz, and Steven Bergner. Tuner: Principled Parameter Finding for Image Segmentation Algorithms Using Visual Response Surface Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1892–1901, Dec. 2011.

[VJC09] Katerina Vrotsou, Jimmy Johansson, and Matthew Cooper. ActiviTree: Interactive Visual Exploration of Sequences in Event-Based Data Using Graph Similarity. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):945–952, Nov 2009.

[VP04] Vivek Verma and Alex Pang. Comparative flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):609–624, Nov. 2004.

[VvdB08] Hans Verhagen and Henk van den Berg. A simple visual model to compare existing nutrient profiling schemes. *Food & Nutrition Research*, 52(10), 2008.

[WAC+13] Nadir Weibel, Shazia Ashfaq, Alan Calvitti, James D. Hollan, and Zia Agha. Multimodal data analysis and visualization to study the usage of Electronic Health Records. In *Proceedings of the 7th International Conference on Pervasive Computing Technologies for Healthcare*, PervasiveHealth '13, pages 282–283, May 2013.

[War09] Matthew O. Ward. Linking and Brushing. In *Encyclopedia of Database Systems*, pages 1623–1626. Springer US, 2009.

[WFR+10] Jürgen Waser, Raphael Fuchs, Hrvoje Ribičić, Benjamin Schindler, Günther Blöschl, and M. Eduard Gröller. World Lines. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1458–1467, Nov 2010.

[WGS07] Alexander Wiebel, Christoph Garth, and Gerik Scheuermann. Computation of Localized Flow for Steady and Unsteady Vector Fields and Its Applications. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):641–651, July 2007.

[Wik16] Wikipedia. Spot the difference. https://commons.wikimedia.org/wiki/File:Globe_and_high_court_fix.jpg, 2016. [Online; accessed 04-April-2016].

[WMK13] Ross T. Whitaker, Mahsa Mirzargar, and Robert M. Kirby. Contour Boxplots: A Method for Characterizing Uncertainty in Feature Sets from Simulation Ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2713–2722, Dec 2013.

[WP09] Andrew T. Wilson and Kristin C. Potter. Toward Visual Analysis of Ensemble Data Sets. In *Proceedings of the 2009 Workshop on Ultrascale Visualization*, UltraVis '09, pages 48–53, New York, NY, USA, 2009. ACM.

[WS06] Jonathan Woodring and Han-Wei Shen. Multi-variate, Time Varying, and Comparative Visualization with Contextual Cues. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):909–916, Sept 2006.

[WSKK06] Yasuhiro Watashiba, Koji Sakai, Koji Koyamada, and Masanori Kanazawa. Visualizing Similarities between Volume Datasets by Using Critical Point Graph and Character Recognition Technique. *The Journal of the Society for Art and Science*, 5(1):11–17, Jan. 2006.

[WT05] Chris Weigle and Russell M. Taylor. Visualizing intersecting surfaces with nested-surface techniques. In *Proceedings of the IEEE Conference on Visualization*, VIS '05, pages 503–510, Oct. 2005.

[WVFH12] Alexander Wiebel, Frans M. Vos, David Förster, and Hans-Christian Hege. WYSIWYP: What You See Is What You Pick. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2236–2244, Dec. 2012.

[YBT01] Méziane Yacoub, Fouad Badran, and Sylvie Thiria. A Topological Hierarchical Clustering: Application to Ocean Color Classification. In *Proceedings of the International Conference on Artificial Neural Networks*, ICANN '01, pages 492–499. Springer-Verlag, Aug. 2001.

[ZCW02] Hualin Zhou, Min Chen, and Mike F. Webster. Comparative evaluation of visualization and experimental results using image comparison metrics. In *Proceedings of the IEEE Conference on Visualization*, VIS '02, pages 315–322, Washington, DC, USA, Oct./Nov. 2002. IEEE Computer Society.

[ZSL+16] Changgong Zhang, Thomas Schultz, Kai Lawonn, Elmar Eisemann, and Anna Vilanova. Glyph-Based Comparative Visualization for Diffusion Tensor Fields. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):797–806, Jan. 2016.

JohannaSchmidt

University Assistant

contact

Krongasse 20/11
1050, Wien
Austria

+43 699 11686240

jschmidt@cg.tuwien.ac.at

languages

native German
fluent English
basic Italian

interests

visualization
medical imaging
visual computing

education

2011-2016 **Doctor** of Technical Sciences — TU Wien, Vienna, Austria
Scalable Comparative Visualization
(expected date of defense: June, 2016)

2006-2011 **Diplom-Ingenieur (Dipl.-Ing.)** in Visual Computing — TU Wien, Vienna, Austria
Interactive Variability Analysis for Initial Sample Testing of Industrial CT Data

2000-2006 **Bachelor of Science (BSc.)** in Media Informatics — TU Wien, Vienna, Austria
Practical Implementation of a Texture Synthesis Algorithm

appointments

2011–2016 **TU Wien** — Vienna, Austria
University Assistant
Research and teaching in the field of visualization at the Institute of Computer Graphics and Algorithms

2007–2011 **AIT - Austrian Institute of Technology** — Vienna, Austria
Software Developer
Management, integration and visualization of heterogeneous biological data (EVOLTREE, EU project)

2005–2007 **ProCom-Strasser** — Strasshof, Austria
System Administrator
Administration of servers, video conferencing systems and the local network; software development projects

skills

Visualization: Scientific visualization, Visual analysis, Information visualization
Programming languages: C++, C, Objective-C, OpenGL, JAVA, JSP, PHP, CSS, JavaScript, AJAX, Python, Perl, XML/XSL, HTML, SQL
Databases: Postgres, MySQL, Oracle
Operating systems: Windows (NT/2000/XP/7/8/10), UNIX (Solaris), Linux (Debian/SUSE/Ubuntu), Cygwin, Android
Tools: Visual Studio (2008/2010/2012/2013/2015), Eclipse, CMake, Hibernate, ITK, VTK, D3, Qt, LaTeX, Git, SVN, Adobe Photoshop, Adobe Illustrator, Blender, MeshLab

teaching

2011–2016 **Visualization I** — TU Wien
Responsible for the lab-work, and the final exams

2015–2016 **Visualization II** — TU Wien
Responsible for lecture units on specific visualization topics

2013–2016 **Computer Graphics** — TU Wien
Responsible for lecture units on specific topics, and the final exams

reviewing

ACE 2012, **CESCG** 2012/2013, **CGI** 2013, **EuroVis** 2014/2015/2016, **IVVAP** 2014, **SIGRAD** 2012/2013, **VIS** 2014/2015/2016, **WSCG** 2014/2015

publications

Johanna Schmidt, Bernhard Fröhler, Reinhold Preiner, Johannes Kehrer, Eduard Gröller, Stefan Bruckner, Christoph Heinzl. “Visual Analysis of Volume Ensembles Based on Local Features” **2016**, no. TR-186-2-16-2. Institute of Computer Graphics and Algorithms, TU Wien, Austria

Johanna Schmidt, Reinhold Preiner, Thomas Auzinger, Michael Wimmer, Eduard Gröller, Stefan Bruckner. “YMCA - Your Mesh Comparison Application” *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, **2014**

Alexey Karimov, Gabriel Mistelbauer, **Johanna Schmidt**, Peter Mindek, Elisabeth Schmidt, Timur Sharipov, Stefan Bruckner, Eduard Gröller. “ViviSection: Skeleton-based Volume Editing” *Computer Graphics Forum* **2013**, vol. 32, no. 3 pp. 461–470

Johanna Schmidt, Eduard Gröller, Stefan Bruckner. “VAICo: Visual Analysis for Image Comparison” *IEEE Transactions on Visualization and Computer Graphics* **2013**, vol. 19, no. 12 pp. 2090–2099

Dieter Kopecky, **Johanna Schmidt**, Silvia Fluch. “Large-scale integration of distributed heterogeneous data sources within EVOLTREE” *Proceedings of the EVOLTREE Conference on Forest Ecosystem Genomics and Adaptation*, **2010**

Jean-Paul Soularue, Frédéric Raspail, Audrey Jacques-Gustave, Christophe Plomion, Thierry Labbé, Dieter Kopecky, **Johanna Schmidt**, Antoine Kremer. “A web portal for oak genetic and genomic data” *Proceedings of the EVOLTREE Conference on Forest Ecosystem Genomics and Adaptation*, **2010**

Johanna Schmidt. “Practical Implementation of a Texture Synthesis Algorithm” *Proceedings of the 9th Central European Seminar on Computer Graphics*, **2005**