Multi-Aspect Visualization:

Going from Linked Views to Integrated Views

Jean-Paul Balabanian

Dissertation for the degree of Philosophiae Doctor (PhD)

Supervised by Eduard Gröller
Institute of Computer Graphics and Algorithms
Vienna University of Technology

University of Bergen
Norway

October 2009

Abstract

This thesis is a delve into the matter of visualization integration. There are many approaches to visualizing volume data and often several of these approaches can appropriately be used at the same time to visualize different aspects. The usual way is to visualize these aspects separately in different views, but integrating the visualizations into the same view can often be the superior approach. We describe the two most used approaches to visualizing several aspects at the same time; linked views and integrated views. We describe some approaches to create integrated visualizations by showing where in the visualization pipeline the integration takes place. We present work produced by the author describing the integrated visualizations developed.

Preface

> "Always be wary of any helpful item that weighs less than its operating manual."
>
> –*Terry Pratchett*

The above quote is a thought that should be in the head of most computer scientists at all times. There are many examples of algorithms and programs that are harder to explain the workings of than the problem they solve. In visualization this is the knife's edge that separates simple to comprehend (good) visualizations from hard to comprehend (bad) visualizations. A visualization should provide more information and better insight than the data by itself but it should be presented in a way that is almost intuitive. This is not always possible so shortcuts, abstractions and exceptions are used to overcome these problems. This usually results in complex descriptions of how the visualization works. Effective visualizations that require simple (or no) explanations should be the goal of every visualization researcher. The work presented in this thesis is hopefully a step in this direction.

Many people were involved in the creation of this thesis, both directly and indirectly, and in this respect the words of Odin comes to mind:

> Ingen så gjevmild
> og gjestmild eg fann,
> han tok ikkje gåvor og takka.
> Eller så gjev-sæl
> med godset sitt,
> han ei lika med løn du takka.
>
> – Håvamål

Thanks for all your help.

Bergen, October 2009
Jean-Paul Balabanian

Contents

Part I

Overview

Chapter 1

Introduction

Scientific visualization of volume data is a vast field of research and many techniques to visualize this type of data has been developed over the years. Examples of these types of techniques to rendering volumes are splatting and raycasting, polygonizing the data using marching cubes or visualizing other aspects of the data such as segmentations, statistics or hierarchies. In some cases it is interesting to convey several aspects of the underlying data at the same time. In this thesis we refer to *aspects* as any data that is co-registered with or abstractly related to the source data. There are many approaches to this but the main contenders are linked views and integrated views. In the end it is the integrated visualizations that create the most compact results and attain the best focus of attention. It is in this area of research that this thesis is contributing.

This thesis is divided into two parts. The work that is the foundation of this thesis is provided in the second part as four research papers. The broader view of the results from these papers are presented in the first part. In Chapter 2 we give a short introduction into volume visualization and the visualization concepts of linked views and integrated views. We also discuss some approaches to integrating visualizations. In Chapter 3 we summarize the results from the papers presented in the second part and relate them to the topics covered in Chapter 2. Finally we conclude in Chapter 4. The results presented in Chapter 3 are taken from the following four papers produced by the author during his PhD project:

Paper I

Sonar Explorer: A New Tool for Visualization of Fish Schools from 3D Sonar Data

Jean-Paul Balabanian, Ivan Viola, Egil Ona, Ruben Patel, and Eduard Gröller

Published in the proceedings of EuroVis 2007

This paper describes a system for processing and visualizing 3D sonar data of fish schools. The volumetric data is visualized seperately as slices and as volume renderings. In addition the system shows the semi-automatic

segmented fish-schools relative to the acquisition path of the research vessel over time as an integrated visualization.

Paper II

Temporal Styles for Time-Varying Volume Data

Jean-Paul Balabanian, Ivan Viola, Torsten Möller, and Eduard Gröller

Published in the proceedings of 3D Data Processing, Visualization and Transmission 2008

This paper describes a technique for merging fully- and partially overlapping volume data-sets that contain several time-steps. Different approaches are used to visualize different aspects of the spatial data. A graphical user interface concept called Temporal Style Transfer Functions is used to interact with the temporal aspects of the data.

Paper III

Hierarchical Volume Visualization of Brain Anatomy

Jean-Paul Balabanian, Martin Ystad, Ivan Viola, Arvid Lundervold, Helwig Hauser, and Eduard Gröller

Published in the proceedings of Vision, Modeling, and Visualization 2008

This paper describes a technique for integrating volume visualization with abstract data. The abstract data is in this case the hierarchy acquired from the anatomical structure of the brain. A graph layout is used to visualize the hierarchical data and the associated volume data is rendered inside the nodes of the graph. Several techniques are proposed that visualize and simplify interaction with the hierarchical nature of the data.

Paper IV

Interactive Illustrative Visualization of Hierarchical Volume Data

Jean-Paul Balabanian, Ivan Viola, and Eduard Gröller

To be submitted

Using the results from Paper III as a foundation several new approaches are presented that illustrate visualization integration where a tighter coupling between hierarchical data and spatial data is demonstrated.

Paper V

$\mathcal{A}$

Jean-Paul Balabanian and Eduard Gröller

Submitted

This paper is a conceptual viewpoint on visualization integration. It describes $\mathcal{A}$-space where visualization algorithms are *points* and integrated visualizations are *interpolations* or *reconstructions* in this space.

Chapter 2

Going From Linked Views to Integrated Views

Visualizing multiple aspects of the same data at the same time is necessary in many application domains. In the following sections we will describe the two main solutions for this problem; linked views and integrated views. Linked views visualize the different aspects separately in several windows or views while integrated views have only one view where all visualizations take place. We will demonstrate some of the possibilities of linked views and integrated views and describe some approaches on how to create an integrated view.

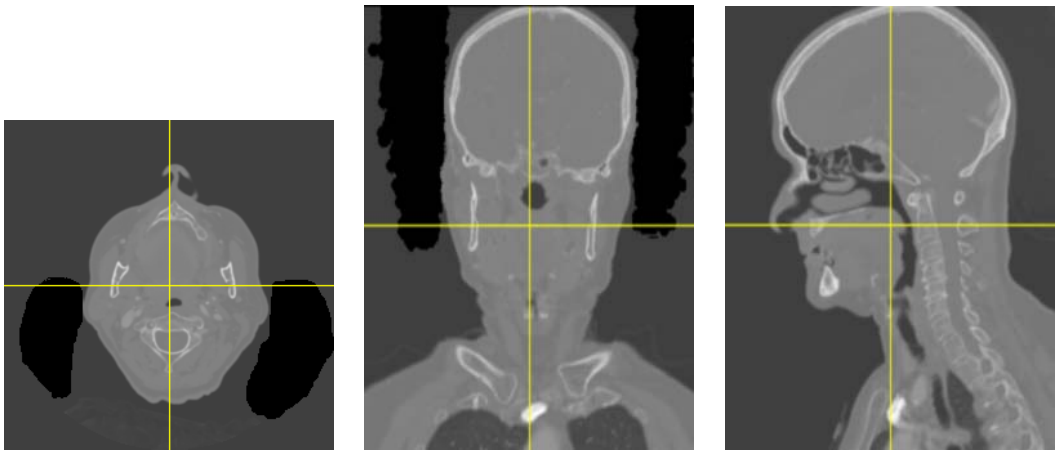
In Section 2.1 we will give an introduction to visualization of volume data. In Section 2.2 we will describe multi-aspect visualization. Finally we will cover different approaches to integrating visualization techniques in Section 2.3.

2.1 Visualizing Volume Data

Visualization is a vast field of research. In this section we will lightly cover some of the principal techniques of volumetric visualization with the main focus on the techniques necessary to produce the results presented in the papers. We will first describe slicing which is a fundamental approach to visualize the raw volumetric data in 2D. Then we will cover transfer functions as a way of manipulating the visual representation output of the covered visualization techniques. Finally we will cover volume rendering and an advanced approach to change the visual representation using style transfer functions.

2.1.1 Volume Data

Volume data is a collection of samples distributed spatially. Each sample is called a voxel. Volumes are usually defined on a regular grid, with a regular resolution in all directions. Sources of volume data are, for example, Computed Tomography



(a) (b) (c)

Figure 2.1: (a) Slicing in the axial (XY) plane (Z-direction). (b) Slicing in the coronal (XZ) plane (Y-direction). (c) Slicing in the sagittal (YZ) plane (X-direction).

(CT) or Magnetic Resonance Imaging (MRI). CT, for example, generates a volume containing density values that represents the density of the scanned tissues. Volume data defined on an irregular grid exists as well, such as 3D ultrasound and 3D sonar data [1]. These sources provide samples on curvilinear grids and generate cones with samples covering larger and larger areas the further away from the source they are.

Several parameters can be acquired at the same spatial location. This type of data is called multi-variate. An example of this type of data is hurricane data where values such as precipitation, temperature, velocity and so on, is available for each voxel.

If the data acquired is recorded over time the result is a time-series. This could for example show how an organ is functioning or the development of a hurricane over time.

Volume data can be classified after acquisition. This is usually called segmentation and is the process of classifying regions of interest into segments. For example in MRI scans the different anatomical regions can be identified and classified so that each individual area can be studied and measured independently.

2.1.2 Slicing

Slicing is the virtual equivalent of cutting a solid object with an arbitrary plane and looking at the object from the inside. Usually the three main slicing directions X, Y and Z are used, which in medical terms are referred to as sagittal, coronal and Axial. In a volume dataset axis-aligned slicing corresponds to retrieving

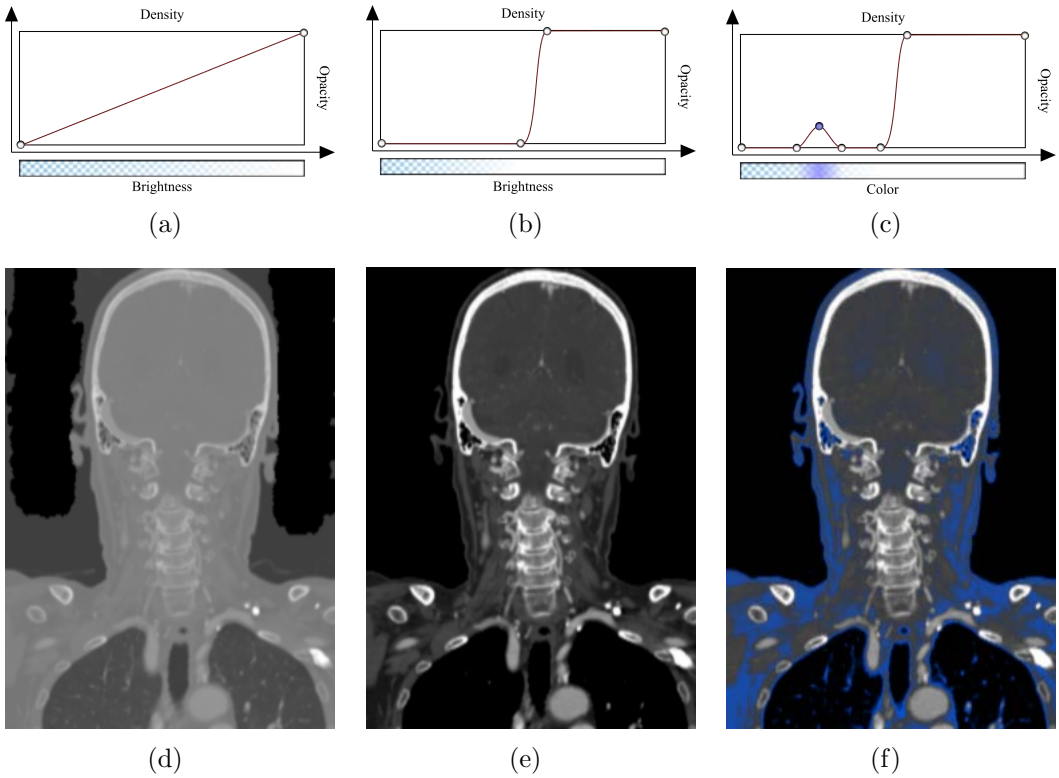


Figure 2.2: (a) The default transfer function mapping one-to-one. (b) A transfer function that effectively removes the lower half of the density values. (c) A transfer function that includes color. (d-f) Slices showing the result of applying the transfer functions (a-c) respectively.

voxels that lie in the same plane as defined by the sample grid. Arbitrary slicing directions are also possible but not as usual since understanding these slices can be much more difficult than the axis-aligned directions.

Figure 2.1 shows the three main slicing directions on a CT scan of a human head. The yellow lines indicate the location of the other slicing planes. In Figure 2.1(a) the horizontal line indicate the location of the slice shown in Figure 2.1(b) and the vertical line indicate the location of the slice shown in Figure 2.1(c).

2.1.3 Transfer Functions

Transfer functions perform a mapping from an input value to an output value. For visualization of volume data this usually means raw or processed volume data as input and a visual representation as output. The visual representation is often defined through color and opacity values. Figure 2.2 illustrates this for medical

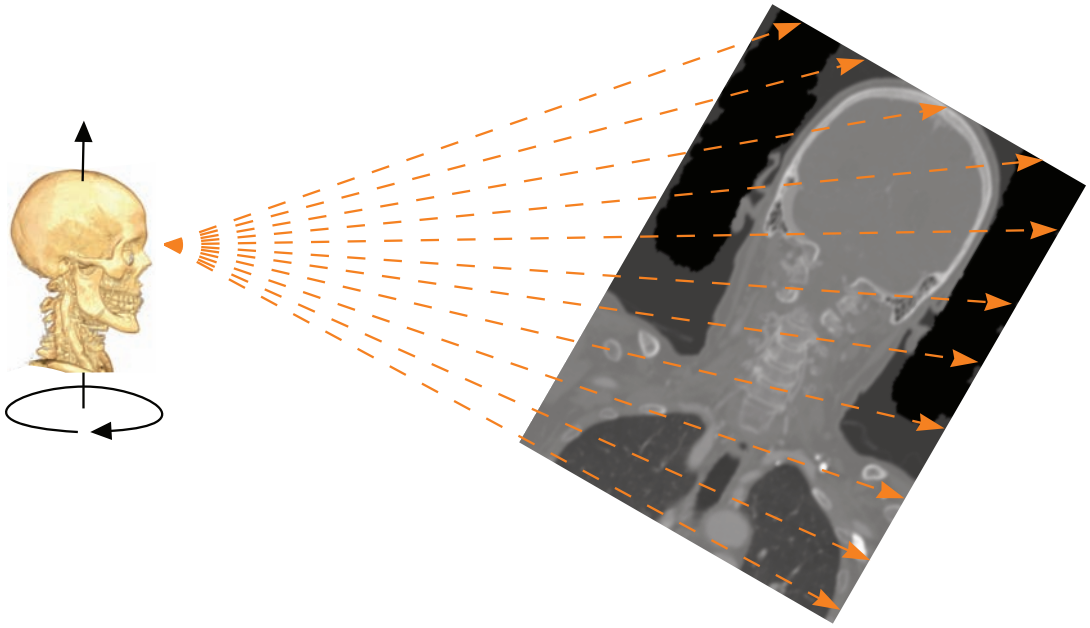


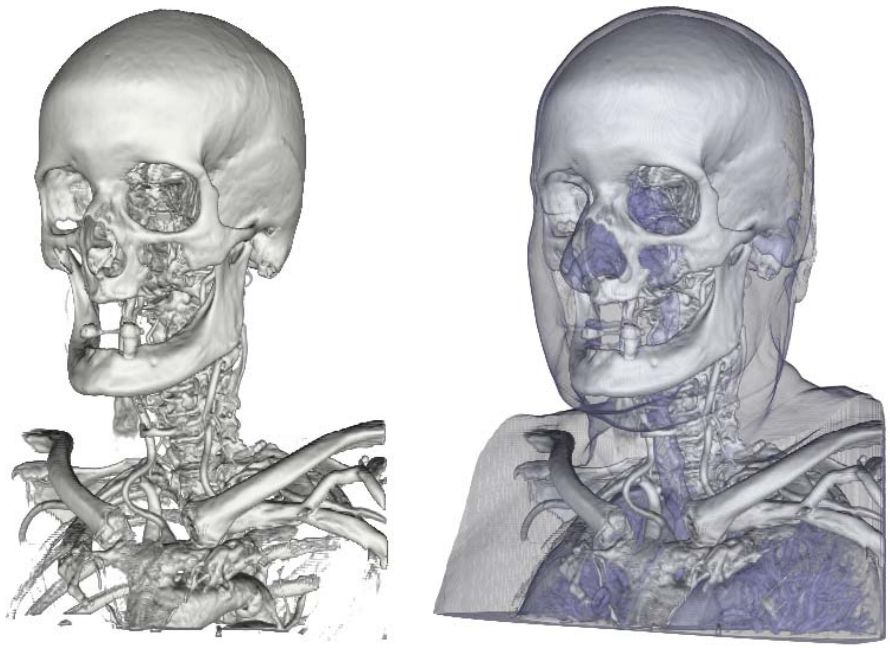
Figure 2.3: Rays are sent from a viewpoint through the volume.

volumetric data. The input data are voxels representing density values from a CT scan of a human head. In Figure 2.2(a) the default one-to-one mapping is shown. Every input value is assigned to an output value and results in all voxels being visible, as seen in Figure 2.2(d). Since manipulating the transfer function changes the opacity of voxels this can be used to emphasize data of interest and deemphasize uninteresting data. The transfer function in Figure 2.2(b) has been adjusted so that only density values above a certain threshold are visible. The effect of this can be seen in Figure 2.2(e) where only tissues with a higher than average density are shown.

A transfer function does not have to be limited to opacity values. In addition to defining opacity, color can also be used. Figure 2.2(f) is similar to Figure 2.2(e) where the high density values are fully opaque, but in addition softer tissue has been slightly highlighted in blue. Several areas can be defined in a similar way. The effectiveness of this, however, depends on the underlying data. In medical data many different tissues have the same density and are difficult to separate using transfer functions only.

2.1.4 Volume Rendering

The visualization of volume data in 3D is heavily dependent on the concept of transfer functions. Since a volume of CT data is virtually an opaque box of densities the necessity of removing obscuring data is obviously important.



(a) (b)

Figure 2.4: (a) Volume rendering using the transfer function from Figure 2.2(b). (b) Volume rendering using the transfer function from Figure 2.2(c).

The technique used in this thesis is raycasting. Raycasting is a technique that sends rays from a viewpoint into a scene, in our case the scene is a volume, and accumulates the data that is intersected along its path. This is illustrated in Figure 2.3. It is with the accumulation that the transfer function comes to use. As the ray accumulates opacity and color values the combined opacity increases. If the opacity reaches the maximum level or the ray leaves the volume, the ray is terminated and a final opacity and color value has been determined for the ray. This opacity and color pair is then assigned to the pixel associated with the ray. The number of rays is usually equal to the viewport resolution defined by the user. The practical implementation of processing the ray is done by stepwise incrementing the ray position and calculating the voxel value for that spatial position. Larger step sizes increase speed but reduce quality and vice versa.

In Figure 2.4 we show two examples of volume rendering which use a raycasting solution based on transfer functions. The volume data is the same dataset used in the slices seen in Figure 2.1. Volume rendering this dataset with the transfer functions shown in Figure 2.2(b) and Figure 2.2(c) results in Figure 2.4(a) and Figure 2.4(b) respectively.

Raycasting is a processor intensive visualization technique and previously considered too slow for realtime visualization. Recent advances in graphics hardware

(a) (b) (c)

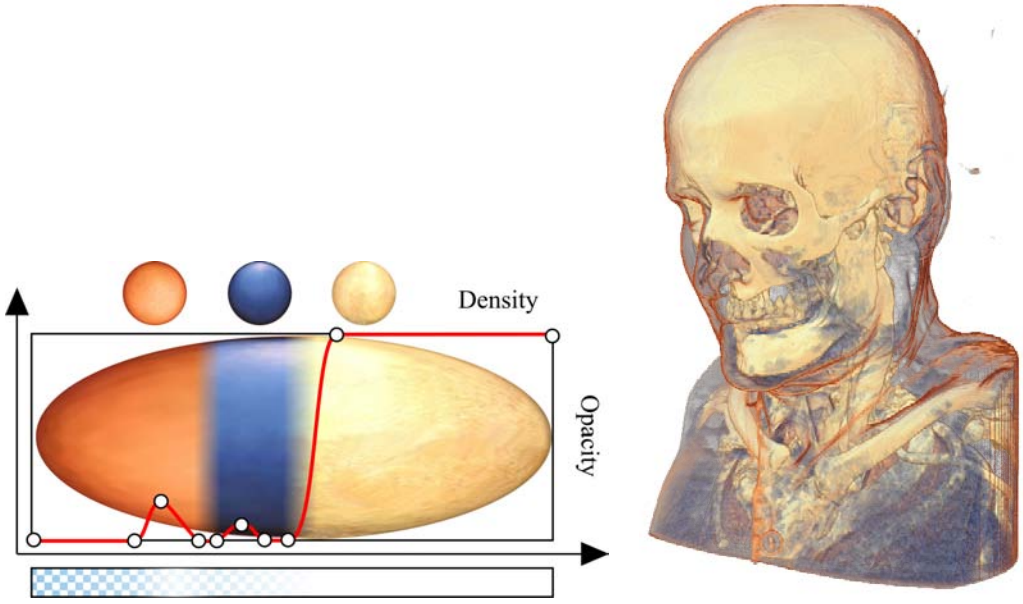
Figure 2.5: (a-c) Volume rendering with a single style applied.

technology and techniques to utilize these advances [9, 15, 16] has made realtime visualization of volume data with raycasting a feasible approach.

2.1.5 Style Transfer Functions

Style Transfer Functions [3] take the concept of transfer functions one step further by changing the visual style in addition to transparency and color. A style is an image of an orthographically projected lit sphere. This sphere can be considered as a simplistic lighting model. Instead of calculating the illumination at a given position based on a normal and light position, the normal is used as a lookup function into the style. This is achieved by projecting the normal on to the style. Figure 2.5 shows the effect of applying a style as a lighting model to the volume rendering. For example the complex purple lighting seen in Figure 2.5(b) is achieved by applying the style from the sphere in the lower left.

Style Transfer Functions incorporate styles into the standard transfer function concept. In addition to changing opacity and color, a style can also change depending on input values. Figure 2.6(a) shows a simple style transfer function where the style also changes based on density values. Where a bone-like style is defined for the high densities, a blue style for medium densities and an orange style for the softest tissues is taken. The result of applying this style transfer function during volume rendering can be seen in Figure 2.6(b) where the skin and respiratory system is orange, muscles and fat is blue and bone and teeth have a bone-like coloring. If the dataset has been segmented the style transfer functions can be applied to individual segments. The result of such an approach can be seen in Figure 2.7.



(a) (b)

Figure 2.6: (a) A style transfer function. (b) Volume rendering using the style transfer function of (a).

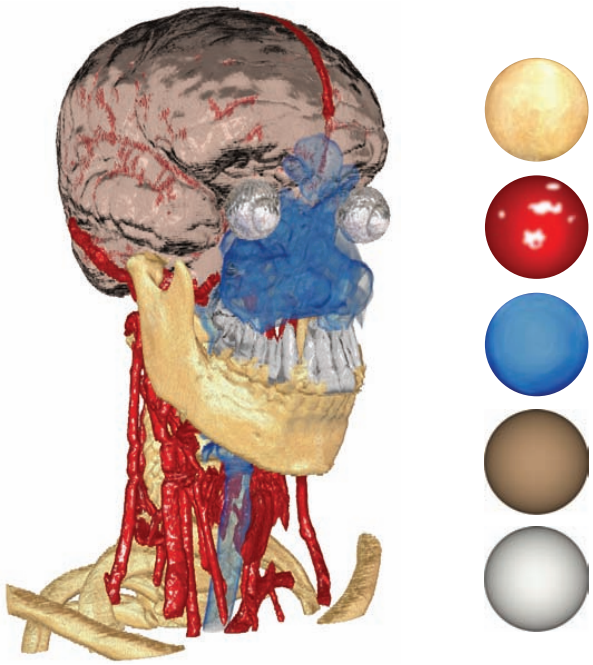
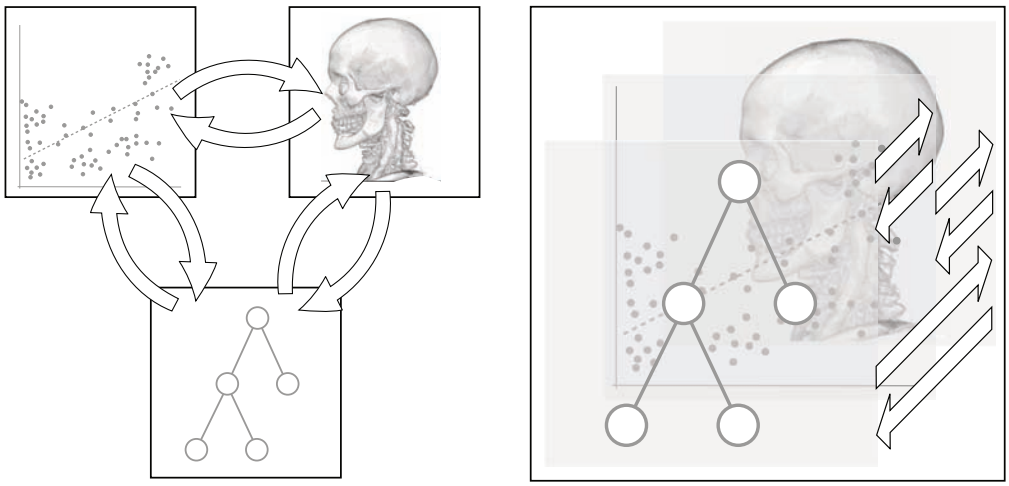


Figure 2.7: Style transfer functions applied to individual segmentations.



(a) (b)

Figure 2.8: (a) The concept behind linked views where several views show different aspects of the data with linking between the views. (b) The concept behind integrated views where several visualizations and linking are integrated into the same image.

2.2 Multi-Aspect Visualization

Multi-aspect visualization is an umbrella definition for many types of visualization approaches. The idea is that a data source that can be multi-modal, high dimensional and with additional abstract information can be visualized in a multitude of different ways. These visualizations themselves can originate from several domains, including scientific visualization and information visualization. Each of the visualizations presents a unique aspect of the data and displaying the different aspects at the same time is termed: *multi-aspect visualization*.

The two main approaches to multi-aspect visualization are linked views and integrated views. Linked views present several separate views or windows with visualizations that cover different aspects of the data and provide a linking mechanism between the views. Integrated views create one single view that combines several visualizations into the same image. The concepts behind these two techniques are given in Figure 2.8. In this section we will describe both of these techniques, starting with linked views in Section 2.2.1 and then integrated views in Section 2.2.2.

2.2.1 Linked Views

Linked views are a Graphical User Interface (GUI) metaphor commonly used in various applications. Google Maps [10] is a typical example of a linked-view

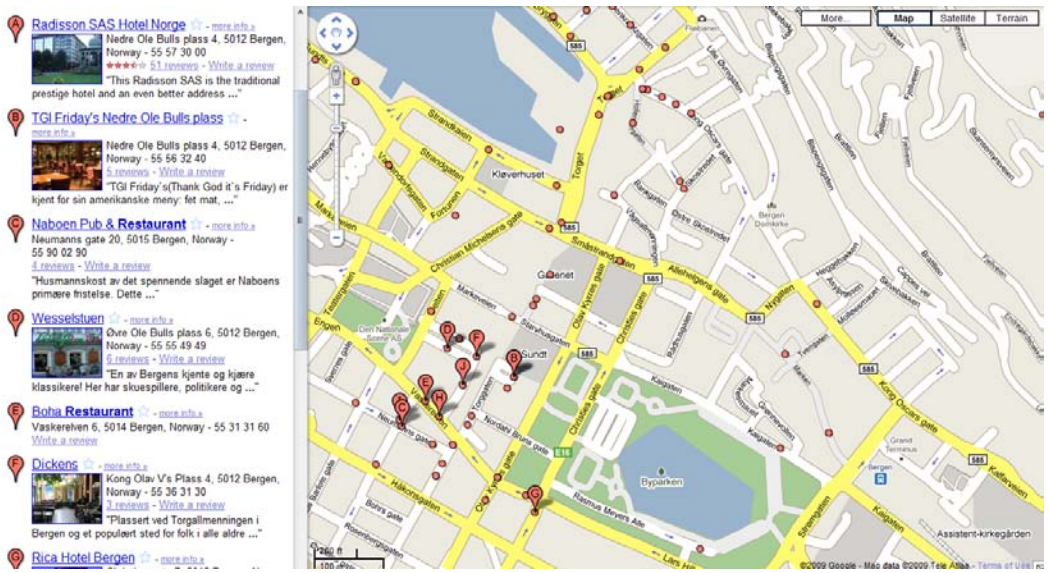


Figure 2.9: Google Maps [10] showing hits for the search query on the left side and the map with result locations on the right side.

setup. Entering a search query in Google maps, for example 'restaurants in Bergen', results in a list of possible candidates. This list is presented vertically on the left side in the browser window. On the right side a map of Bergen is shown with markers for all the candidates. The first ten hits are labeled with A-J. Figure 2.9 illustrates this. Clicking on one of the results on the left side focuses the location of that restaurant and pops up a balloon containing more information in the map view. This is conceptually the same as the linked-views approach used in scientific applications.

A linked view consists of at least two windows or views of the same data but visualizing different aspects of it and at least one link between these views. Figure 2.10 is an example of this concept. In this Figure three of the windows visualize the data as slices and the fourth view as a 3D volume rendering. The different slicing-plane locations are indicated both on the slices and in the volume rendering, as red, green and blue lines. The colors represent sagittal, coronal and axial slicing directions respectively. The linking between the views concerns the current slice position in all views. Changing a slice position, for example in the axial direction, would update the blue line in the sagittal and coronal slice views in addition to the volume-rendered view. The blue line indicates in all affected views the position of the slice in the axial direction. There is no linking from the volume-rendered view to the slice views in this setup (although there can be).

An advanced version of this type of setup is LiveSync [14]. LiveSync is a system that basically looks like the typical slice + volume-rendered view setup but has advanced linking capabilities. If the user selects an interesting area in a slice,

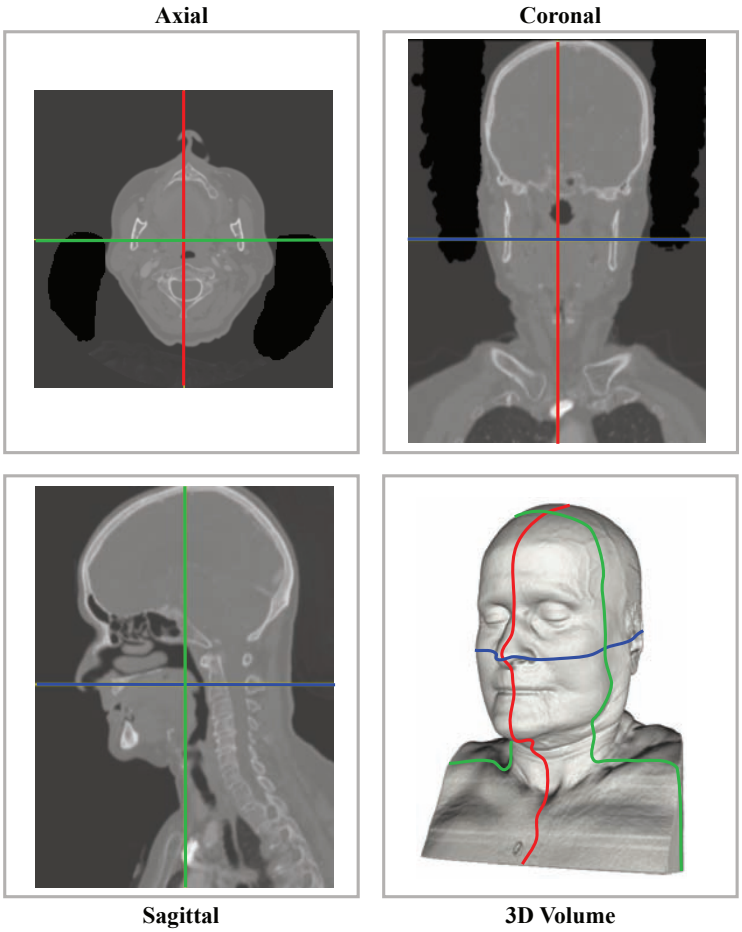
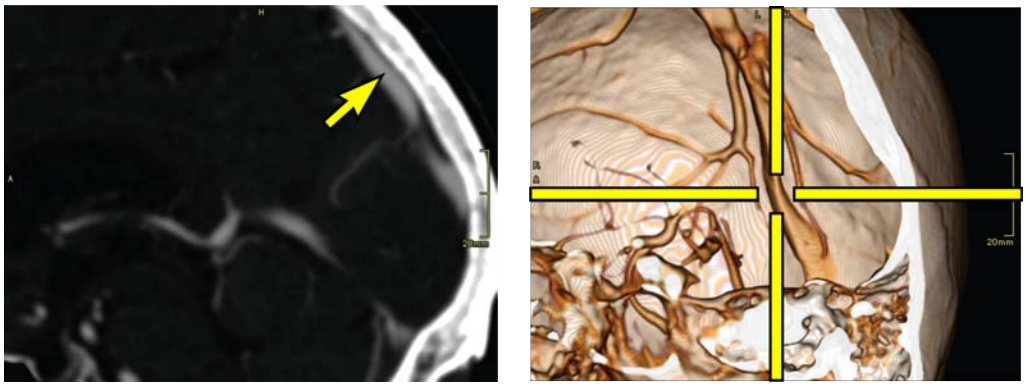


Figure 2.10: A basic linked-view setup.



(a) (b)

Figure 2.11: (a) The sinus vein is picked in a slice. (b) An unoccluded 3D view of the vein is automatically generated by LiveSync [14].

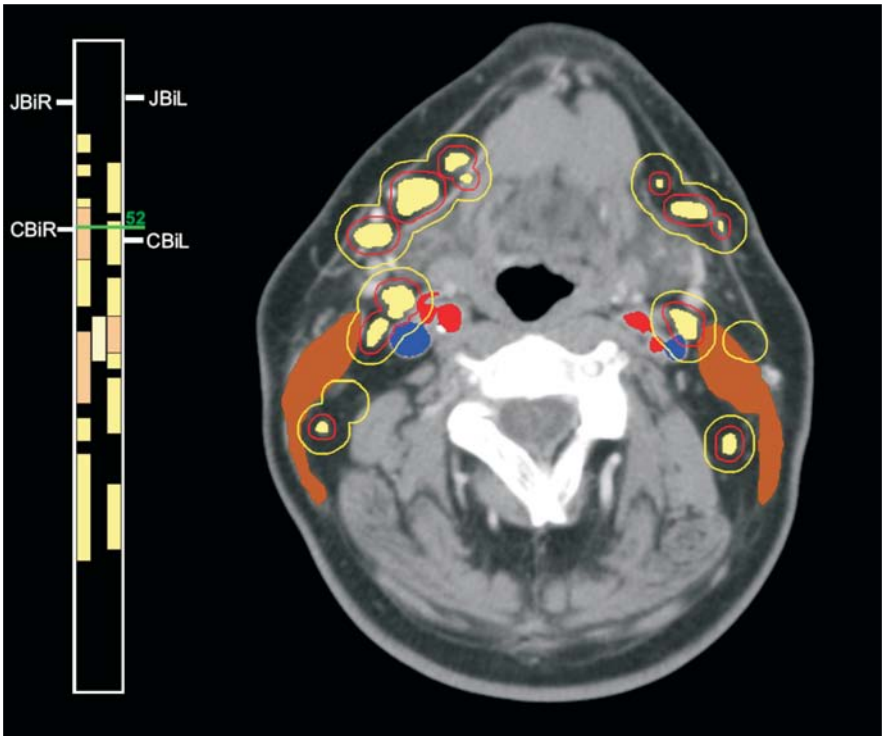


Figure 2.12: The LiftChart [19] on the left indicate the extent of segments in the orthogonal direction to the slicing plane shown on the right. The slice on the right show the location of segments and additional information.

the optimal viewing position and direction is calculated and the camera is moved to that position. From this updated viewpoint several procedures are done automatically. A new transfer function is calculated that highlights the interesting region and a clipping plane is moved to remove occluding elements. Figure 2.11 illustrates this interaction metaphor with the user selecting an interesting region in a slice, the sinus vein, in Figure 2.11(a), and the updated 3D view being shown in Figure 2.11(b).

Scientific data often has associated information that has an abstract nature. For example the volume data can be segmented and the segmentations can be labeled or ordered in hierarchies. In the simplest case a list of segments is shown in one view, the 3D volume rendering in another view and the segments can be turned on or off. LiftChart [19] is another example where one view shows a slice with several segmented tumors and lymph nodes while another view shows the extents of the different segments in the slicing direction. Figure 2.12 displays this with the LiftChart on the left side of the image.

Instead of a list of segments a tree view can be used to show the hierarchical nature and relationship between segments of the data. Figure 2.13 gives an illustration of how this could be implemented. On the left side a hierarchy describing

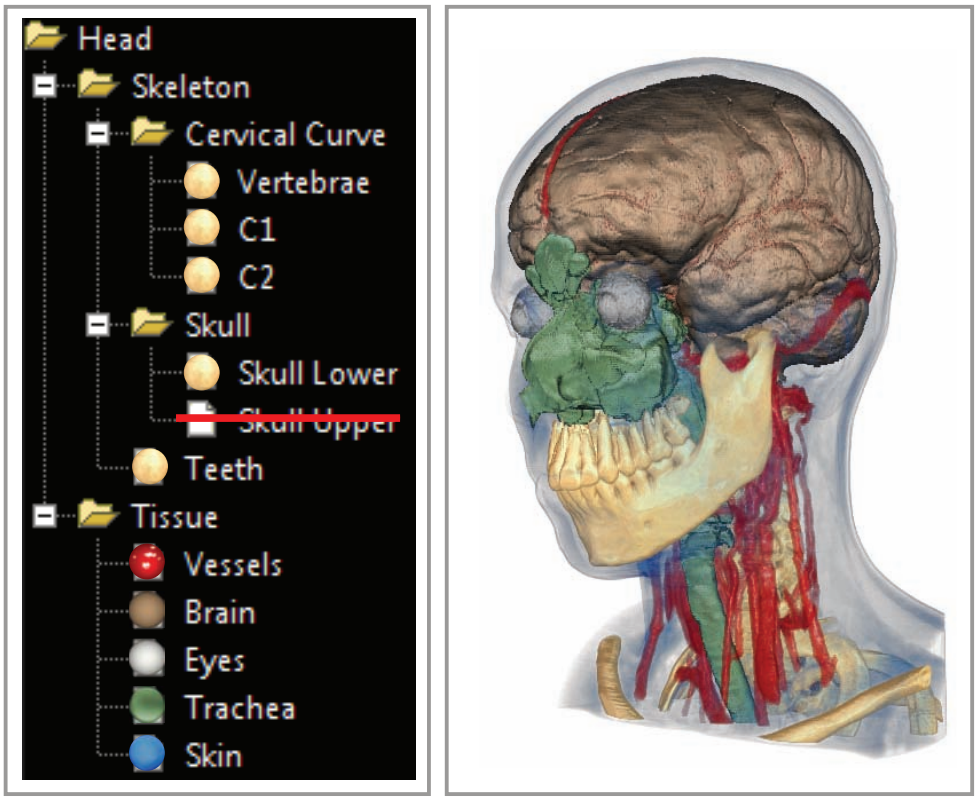


Figure 2.13: A basic linked-view setup.

the grouping of the segmentations is shown, in addition the tree view displays the styles that have been applied and whether a segmentation has been turned off or not. The image resulting from manipulating the tree view is shown on the right side.

Another example of abstract data is statistical data. Statistical data can be collected from a multitude of sources and can be associated with the volume or each voxel. For example the WEAVE system [11] or SimVis [8,7] both do linking between statistical data and volume renderings. In SimVis a scatterplot shows the relationship between two interesting parameters. A pattern in the scatterplot is identified and the interesting region is brushed. The brushing results in updated visualizations of other statistical views but also in modified volume renderings. Figure 2.14 is a screenshot of SimVis in action. Two scatterplots on the left are brushed to produce the result seen on the right.

A characteristic with linked-view setups is that if one is working with datasets of medium complexity the number of views is moderate. An example setup could for example be slicing windows, volume rendering and scatterplots which result in a manageable number of views. If on the other hand one is working with high complexity data, with several modalities, multiple attributes and abstract

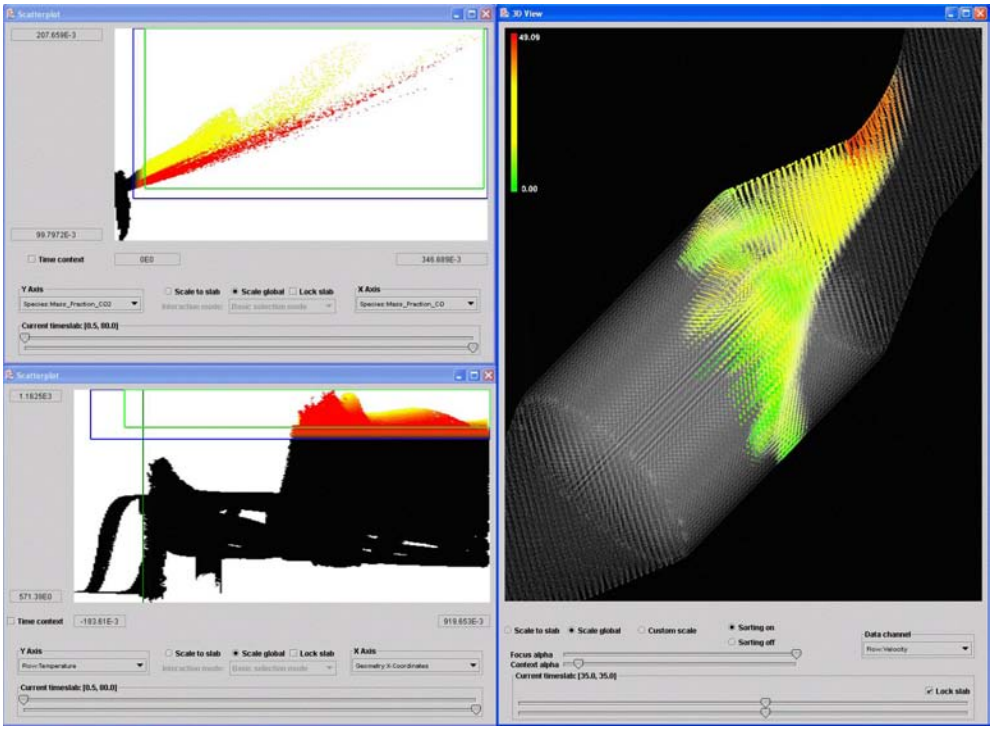


Figure 2.14: SimVis [17] in action. On the left two different scatterplots brushed to create the volume-rendering on the right.

data the number of views increases considerably. The increased number of views results in increased shifting of focus of attention and in tracking changes between views. Another problem is the increased complexity of linking between views. For example changing a parameter in one view could result in simultaneous changes in many other views. It would be increasingly challenging to track all of these changes. Many of these issues can be alleviated significantly with integrated views.

2.2.2 Integrated Views

Integrated views are visualizations where all the information and interaction from a linked-views setup is presented in a single view. A simple example of an integrated view is Intellicast [13]. Intellicast is an online service that integrates weather data with a geographical map. In Figure 2.15 an example image is given. This image shows the weather situation on the 28th of August 2009 on the east coast of the USA. The lowest layer of the integrated visualization is the topographical map of the area. Integrated on top of that is a map depicting roads, place names, and state and county borders. The top most level in this map is the precipitation radar data. These are depicted as the green, yellow, and red

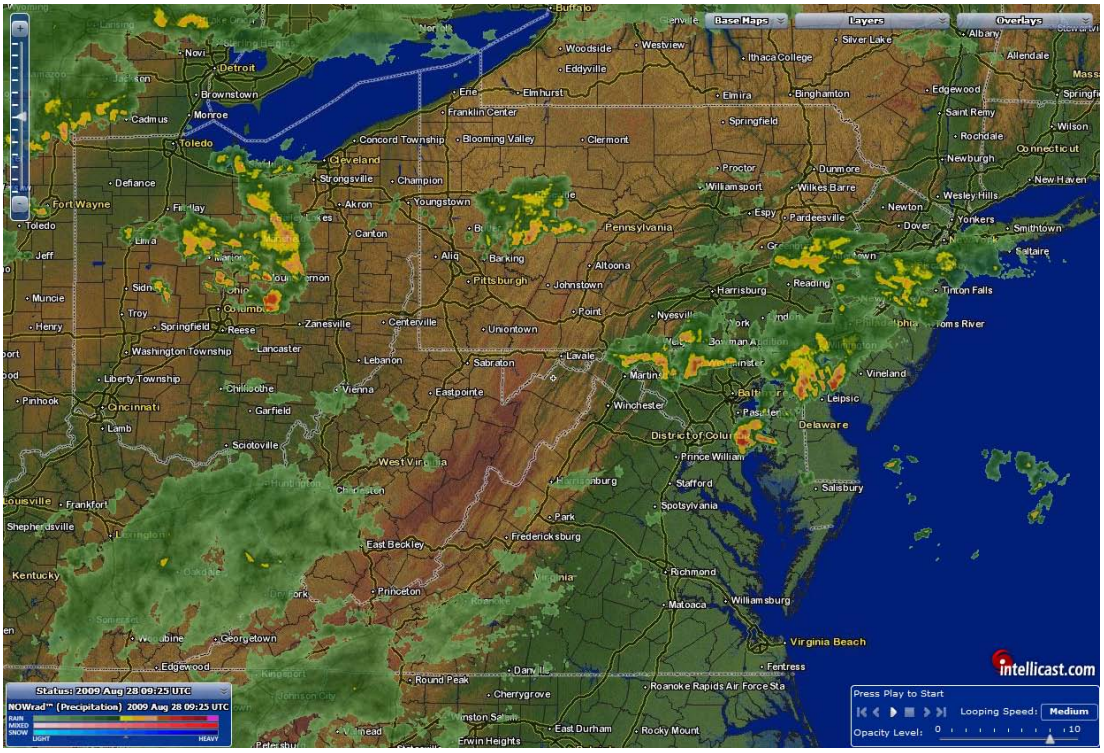


Figure 2.15: Weather data visualization from Intellicast [13].

clouds indicating the amount of precipitation detected. The Intellicast system provides the possibility to show other types of data than precipitation, for example satellite-captured clouds or temperature.

In some cases, going from a linked-view to an integrated-view can be reasonably straightforward. The linked view with slicing and volume rendering shown in Figure 2.10 contains a simple spatial relationship between slices and the volume data. Deriving an integrated version can be to simply insert a slice into the volume rendering at the appropriate location. The result of such an approach can be seen in Figure 2.16 where the sagittal slice has been integrated into the volume rendering. The spatial relationship between the slice and the volume is now trivial to comprehend.

VolumeShop [2] is a visualization framework with several visualization techniques available. The framework simplifies the work necessary to create integrated visualizations. The illustrative visualization shown in Figure 2.17 was created in VolumeShop and depicts a labeled anatomical view of a carp with a close-up of the swim bladder. The approach used to create this image is based on a CT-scanned carp where the swim bladder has been segmented. The swim bladder is given a different visual style than the rest of the structures. In addition a close-up is rendered alongside the carp. Several anatomical parts have been labeled

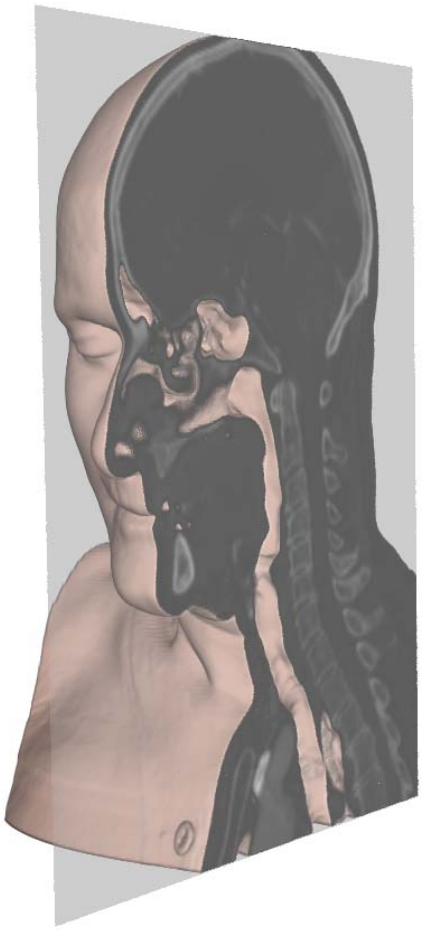


Figure 2.16: Volume rendering with an integrated sagittal slice.

and are positioned in such a way that they do not occlude other structures. The integrations in this visualization happens at several stages. The segmentation, for example, is performed at the data stage. The incorporation of the labels, though, happens at the render stage. This process needs to spatially locate a non-occluding position for a label and needs to use the results of the volume renderer to be able to perform this.

Figure 2.18 shows a visualization concept, proposed by Termeer [18], where many aspects are visualized within a single image. A model of the heart and the arteries has been generated. The rendering of the blood vessels shows which artery is responsible for providing flow to a specific area. The surface of the heart is color coded based on the amount of blood supply measured. Dotted white lines indicate regions of equal supply. A 2D plot, called bulls-eye plot, at the bottom shows the heart surface unwrapped. For a selected area the yellow arrows indicate the most likely contributing arteries with the relative contribution indicated in

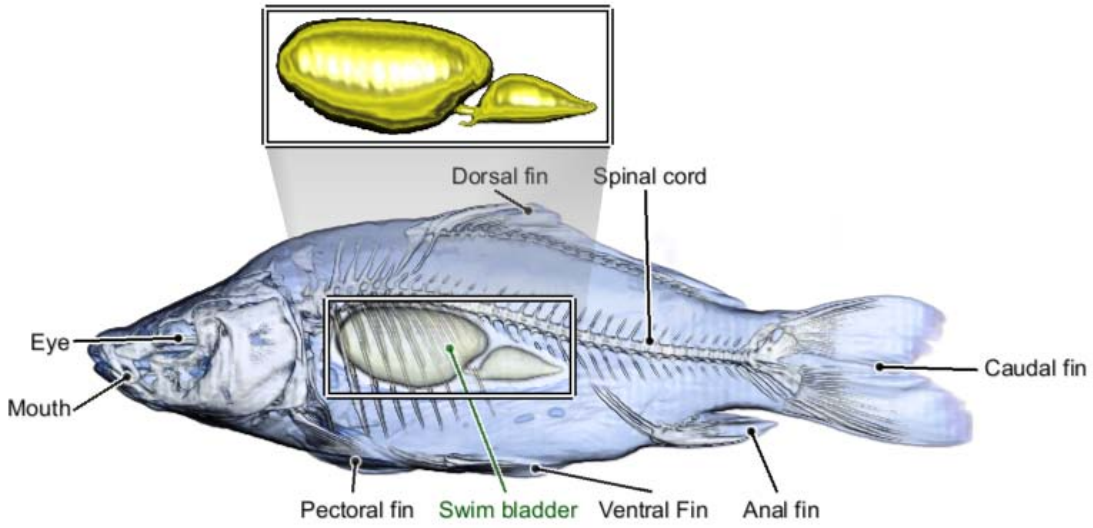


Figure 2.17: An illustrative rendering created with VolumeShop [2] showing a carp. Labels indicate the location of several features and the swim bladder is shown enlarged.

the width of the arrows. The integration in this visualization happens at several stages. The heart and artery models depend on segmentations performed at the data stage while incorporating the blood flow and the yellow arrows onto the heart surface is performed at the render stage.

A technique called MIDA [4] integrates Direct Volume Rendering (DVR) and Maximum Intensity Projection (MIP) into a visualization that preserves the complimentary strengths of both techniques, i.e., efficient depth cuing and parameter less rendering. Since some datasets look better with DVR and others are best viewed with MIP, MIDA lets the user interpolate between DVR and MIP. In Figure 2.19(a) only DVR is used and in Figure 2.19(c) only MIP is used while Figure 2.19(b) shows the resulting MIDA visualization of the dataset. The integration in this visualization happens at the render stage. It is the MIDA visualization algorithm that performs the interpolation between the two different techniques during raycasting.

One of the clear advantages of integrated views is that the focus of attention is kept in one view. It is not necessary for the user to shift focus between different views to keep track of changes. Another benefit is in the case of several sparse views where the available image space can be more efficiently utilized. An example of this are the labels inserted into the carp rendering in Figure 2.17. The space used by the labels is not required by the other structures except for the label arrows. If the user wants to see some detail hidden under one of these lines then the arrows would have to be removed.

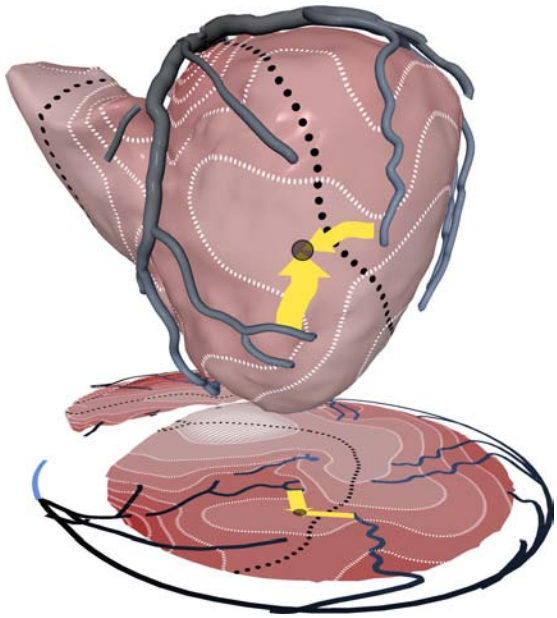
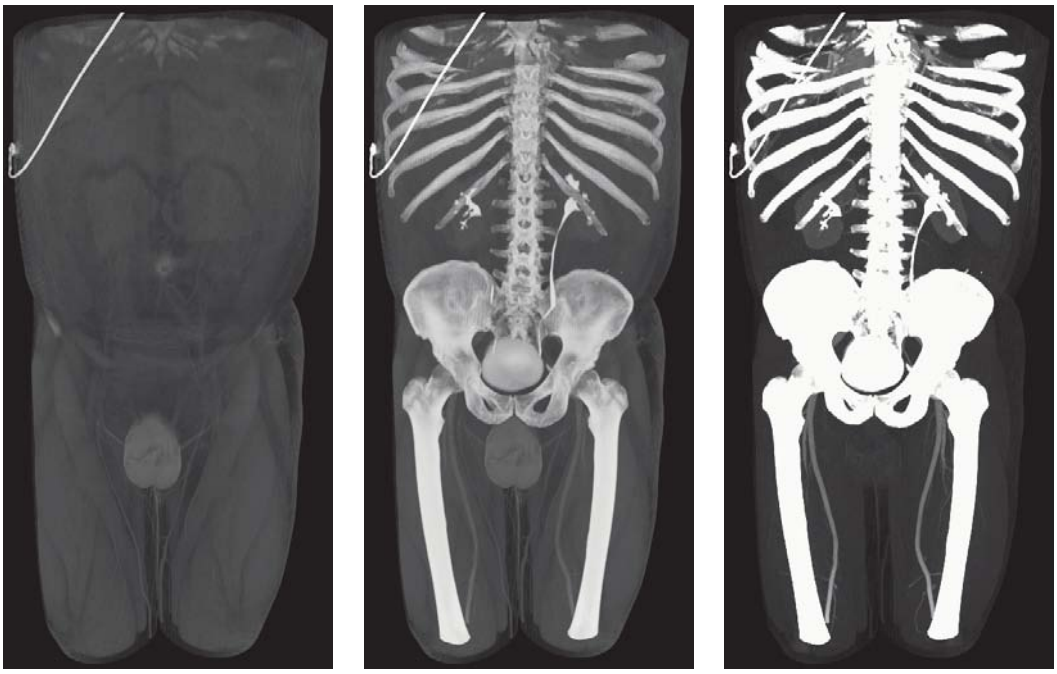


Figure 2.18: Anatomical rendering of the heart with an integrated bulls-eye plot. [18]



(a) (b) (c)

Figure 2.19: Full body CT angiography rendered using (a) DVR, (b) MIDA [4], and (c) MIP.

2.3 Integrating Visualizations

Integrated views is figuratively speaking to take the separate visualizations from linked views and combining them into one view. The process of creating an integrated visualization is not simply combining them, there are several issues to consider. Integrating visualizations can be done at different stages of the visualization pipeline, which we will discuss in Section 2.3.1. Two typical problems of visualization integration are also discussed. The first is occlusion handling which we discuss in Section 2.3.2 and the second issues information overload discussed in Section 2.3.3.

2.3.1 Approaches to Integration

The integration of different visualizations can be classified according to where in the visualization pipeline it takes place. Bürger and Hauser [5] proposed a pipeline for multi-variate visualization that can also be applied to multi-aspect visualization. The visualization pipeline is divided into three different stages and it is at these stages that integration of visualizations can be performed. The three stages in this pipeline are first the data stage, second the rendering stage and finally the image stage. A simplified version of the visualization pipeline they proposed can be seen in Figure 2.20.

Figure 2.20: Simplified visualization pipeline.

The first stage is at the data level, where integration corresponds to data merging, processing and filtering. This stage generates a new or transformed data-set for visualization. Data-set merging can happen at different integration levels. For example co-registering different modalities is at one end of the possibilities, where the data can still be separate but have a common frame of reference (low integration level). A high integration level means that a completely new merged data-set is generated. An example of this are Chronovolumes [20]. Chronovolumes is a technique that takes a volume data-set with several time-steps and applies temporal operators. The result is a single volume where the behavior in time is condensed and this volume is used as a basis for visualizations.

The second stage is the rendering stage. Integrating at this stage could for example be a visualization that takes two different registered modalities such as CT and MRI and during rendering would generate different types of visual representations based on some threshold factors. For example the CT could be

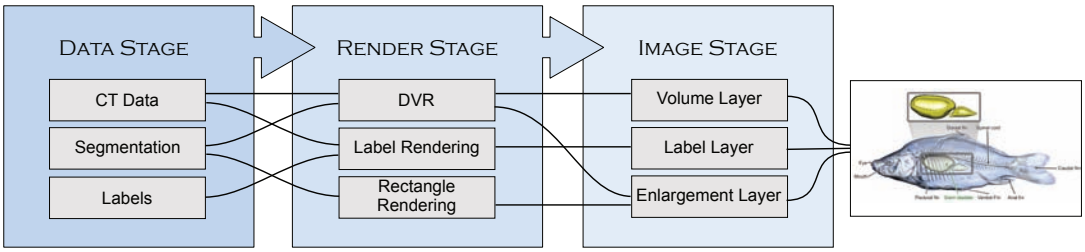


Figure 2.21: Visualization pipeline showing the different integrations necessary to create the image in Figure 2.17.

selected for visualization if the intensity in the CT data is high and the corresponding value is low in the MRI. Another example of integration at this stage could be the integration between rendering techniques. During rendering different visualization techniques are used to visualize different aspects. The selection of techniques can be based on spatial location, threshold values or other parameters. MIDA is an example of this type of integration.

The final stage is the image stage. At this stage the image results of different visualization techniques are merged into one image. A simple example for this approach are layer based visualizations such as Google Maps and Intellicast. The different layers are simply merged together using compositing techniques where transparent areas on one layer lets you see through to the next layer. Obviously the ordering of the layers has an impact on the resulting image.

Creating a complete integrated visualization will usually result in several integrations at several stages. For example to create the result seen in Figure 2.17 several integrations took part. The bottom layer is the volume rendering of the carp. This volume rendering considered the segmented swim bladder differently and presented it with a different visual representation than the rest of the carp. The next layer contains the labels where the location of the presented anatomical parts needs to be considered in 3D relative to the viewpoint of the volume rendered carp. The final layer is the enlargement of the swim bladder. The swim bladder is rendered separately inside a rectangle and the originating rectangle needs to be positioned appropriately relative to the orientation of the carp. The whole process is summarized in Figure 2.21.

2.3.2 Occlusion Handling

Occlusion may occur when two different visualizations try to show a visual entity at the same pixel. If one visualization completely covers another one then it is necessary to highlight the important information in that area. There are many approaches to solve this problem. A typical technique is to use opacity where a visualization on top of another one is rendered semi-transparently. In

another approach a threshold value dependent on the underlying data decides which visualization to use. For example the work by Burns et al. [6] proposes an importance driven approach to decide which features can occlude others. A tracked ultrasound slice is integrated with CT data, the straightforward approach is to remove data in front of this slice but this would also remove important structures. Their solution is to give different organs different importance values. In this way important features are still visible in front of the ultrasound slice.

2.3.3 Information Overload

Another issue that can occur in integrated views is information overload. Information overload is the issue of presenting more information than the user is capable of handling. This happens when visualizations are showing very large data-sets that could be of interest but no measures have been taken to restrict the amount of information presented. A simple example of information overload is a geographical map showing several cities where every city, village, intersection and mountain has been labeled. In addition every road is shown, which means every type of road from highways to gravel paths are marked and named. This scenario would lead to thousands of labels and small lines showing roads, making it impossible to discern anything of value. A possible solution to this would be to adapt the visualization to the scale at which the map is viewed which is what most map applications are doing to prevent information overload. So while looking at the map from an overview perspective, only city names and highways are shown but if you zoom in more space is available and details of lower importance can be provided.

Another approach is to use the Focus+Context [12] concept to reduce the amount of information presented. The idea is to show the most important information in high-detail and the context of what your are seeing in low detail. A simple example of this approach is to use color saturation so that the object in focus is fully saturated while the context is desaturated.

2.4 Getting There

The task of creating an integrated visualization does not have a general solution. It is possible, though, to disassemble the integration process into smaller visualization tasks. The disassembly should be in line with the visualization pipeline discussed earlier. This means identifying at what stage integration should be implemented. With all of the building pieces determined it should be possible to predict if occlusion or information overload will be a problem. Solving the occlusion and information overload problems for smaller tasks improves the chances of being able to integrate additional visualizations. This approach will simplify the

creation of an integrated visualization but will not guarantee the achievement of a perfect solution.

In some cases there may not be a satisfactory solution to the integration problem at all. For example we consider the slicing solution shown in Figure 2.16: if the goal would have been to show all three slicing planes at the same time this approach would lead to a more cluttered solution. The resulting visualization would include three slicing planes intersecting a box, leaving little 3D data information left to be shown and slices of reduced size.

Integrated visualizations have been around for a long time and will in the future become more important as the complexity of data increases. With the massive increase in data-set size and complexity new types of integration will be necessary in the future. Systematically categorizing and relating visualization algorithms may be necessary to simplify the process of creating these new integrated visualizations. A system like this would help in identifying what type of visualizations are suitable for the available data or indicate missing algorithms.

Chapter 3

Results

In this chapter we discuss the five papers included in the second part of this thesis. The aim of the discussion is to present the results from these papers in a visualization integration context. The first paper to be presented is *Sonar Explorer: A New Tool for Visualization of Fish Schools from 3D Sonar Data*. The paper basically describes a linked-view setup but one view is an integrated view as well. The next paper is *Temporal Styles for Time-Varying Volume Data*. This paper describes a visualization that condenses a time-series into a single image. The two following papers, i.e., *Hierarchical Volume Visualization of Brain Anatomy* and *Interactive Illustrative Visualization of Hierarchical Volume Data*, integrate a multitude of visualization techniques for hierarchically defined volume data. The last paper is $\mathcal{A}$ where we describe a conceptual space for visualization integration.

3.1 Sonar Explorer

Sonar Explorer is a system for exploration and navigation of 3D sonar data. A research vessel acquires tracking information in addition to one volume per second. This results in a data-set with partially overlapping volumes over time. The goal of the acquisition is to monitor fish schools. The linked views from the resulting application are shown in Figure 3.1. The top row of the application shows horizontal slices, vertical slices and a volume rendering of the current time-step. The center row shows an integrated visualization of the vessel path, vessel location and fish school. The bottom row shows several time-steps around the current time step of the volume data from a top view.

The integrated visualization in the center row is in this context the interesting one. This view shows the path the research vessel took during acquisition including position and angle offsets caused by waves. The current volume of the current time-step is highlighted with a lighter shade of gray. In addition the semi-automatically segmented fish school has been integrated into this visualization.

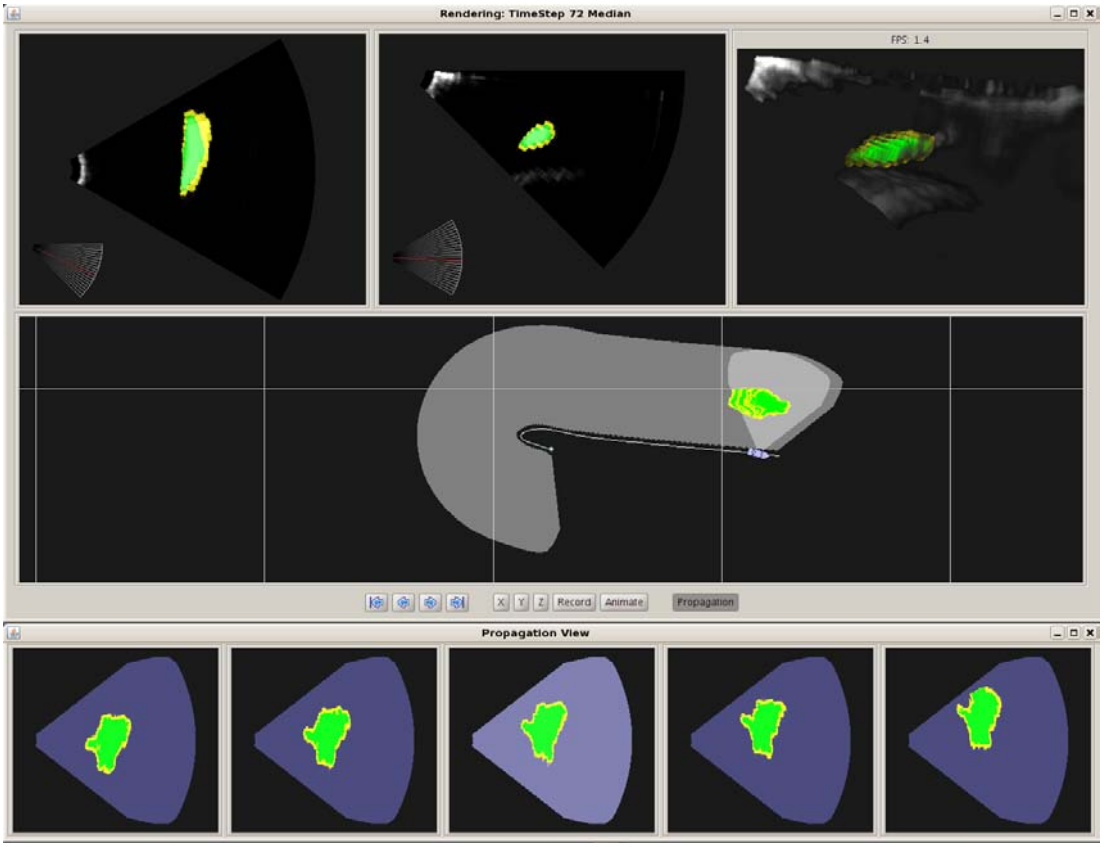


Figure 3.1: Sonar Explorer application showing a linked-view setup. Top row: horizontal slices, vertical slices, volume rendering. Center row: integrated visualization of vessel path and fish school location. Bottom row: several time-steps of the volume data.

This visualization provides the user with a lot of information at a glance: the path of the vessel, the location of the selected time-step and the shape and location of segmented fish schools are all there. Adding a nautical map with depth information as a basic layer would further increase the expressivity without information overload. A problem with this visualization is that over several time-steps the fish school does not move much and so the overlapping areas occlude each other. A solution to this is presented in the paper on *Temporal Styles for Time-Varying Volume Data*.

3.2 Temporal Styles

Temporal style transfer functions (TSTF) is the concept of extending style transfer functions into time. The horizontal axis in a transfer function typically represents density, however it can instead represent any temporal aspect of a time-series

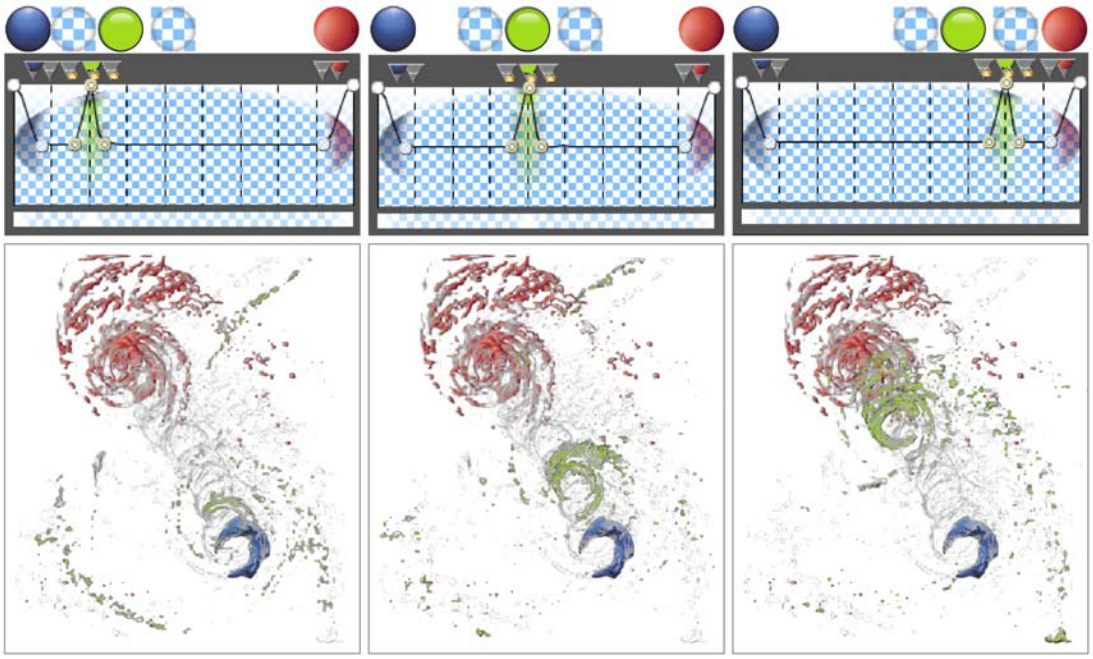
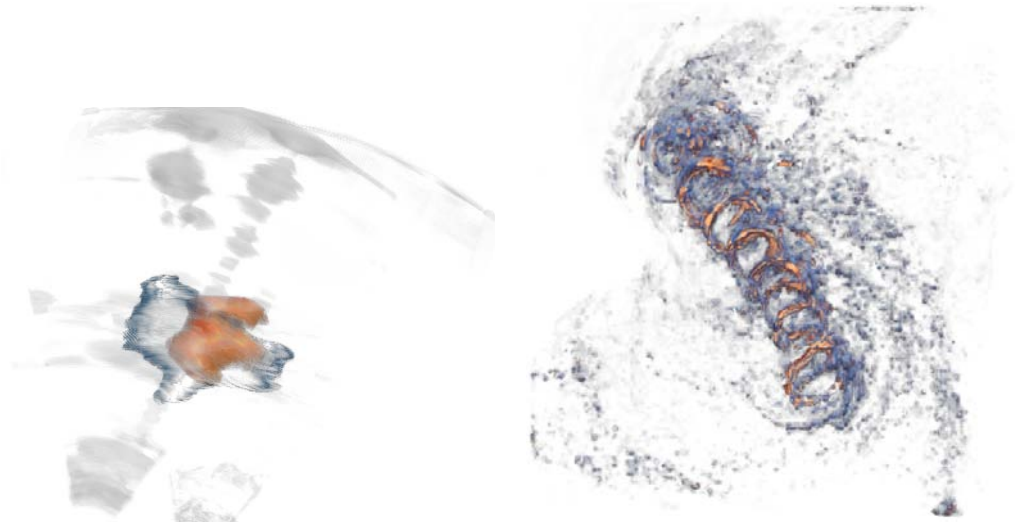


Figure 3.2: Time-step based temporal style transfer function on the hurricane Isabel data-set. First time-step blue, last time-step red and from left to right in green: time-step 3, 5 and 8.

data-set. For example the horizontal axis can refer to individual time-steps with the effect of applying a style for a specific number of time-steps only. Figure 3.2 illustrates this technique. The data-set visualized is the hurricane Isabel with 10 time-steps. Three TSTFs are shown at the top of the figure. For example the left one highlights the first time step in blue and the last one in red. The third time step is assigned a green highlight while the intermediate time-steps are given a semi-transparent cloudy effect. The result of applying this TSTF to the hurricane dataset is shown in the left most volume rendering. Moving the green highlight in the TSTFs results in emphasizing temporally successive structures in the volume renderings.

The same technique can be applied to sonar data. Figure 3.3(a) shows the result of setting the first time-step to orange and the last time-step to a blue outline. From this visualization we can conclude that the mass of the fish school is spreading and that it is not moving in any specific direction. The resulting image conveys more information about the temporal behavior as compared to Figure 3.1.

Another approach of condensing the temporal aspect is to represent density change in time instead of time on the horizontal axis of the TSTF. The TSTF in this case will represent low density changes on the left side and high density changes on the right side. Figure 3.3(b) shows the result of defining such a TSTF



(a) (b)

Figure 3.3: (a) Result of applying a TSTF with the first time-step in orange and the final time-step in a blue outline. (b) Hurricane Isabel with a TSTF that highlights areas of high density changes in orange and low density changes in blue.

where low changes are defined as blue and high changes as orange.

The integration required for this type of visualization is performed at the render stage. During visualization a temporal operator is applied at every spatial position which then condenses the temporal information and generates a visual representation for it. A drawback of this type of integration is the problem of information loss. If the number of time-steps overlapping at a spatial location is high and the temporal behavior is not easily defined then creating a visual representation is quite a challenge.

3.3 Hierarchical Volume Visualization

The next two papers showcase several techniques for integrating hierarchical information with segmented volume data. The main idea is illustrated in Figure 3.4 where a small subset of a hierarchy is shown. In this image a graph is used to describe the hierarchy, with circles representing the nodes and lines showing relationships. Inside the nodes corresponding subsets of the volume data are rendered. The cervical curve is shown on the right side of Figure 3.4. Color coded segments indicate the location of the C1 and C2 vertebrae and the other vertebrae. The cervical curve's location in the skeleton node is shown as the red structure with outlining indicating occlusions. In the leftmost node the head is

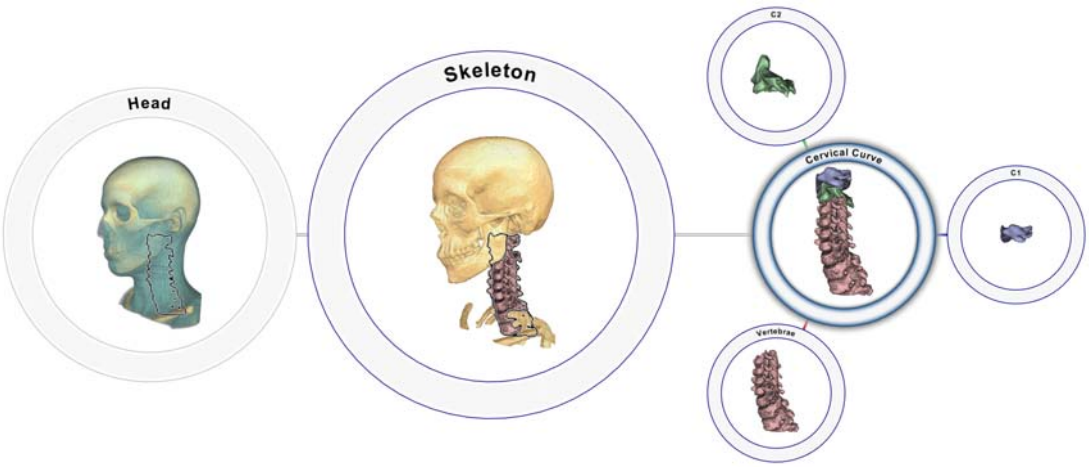


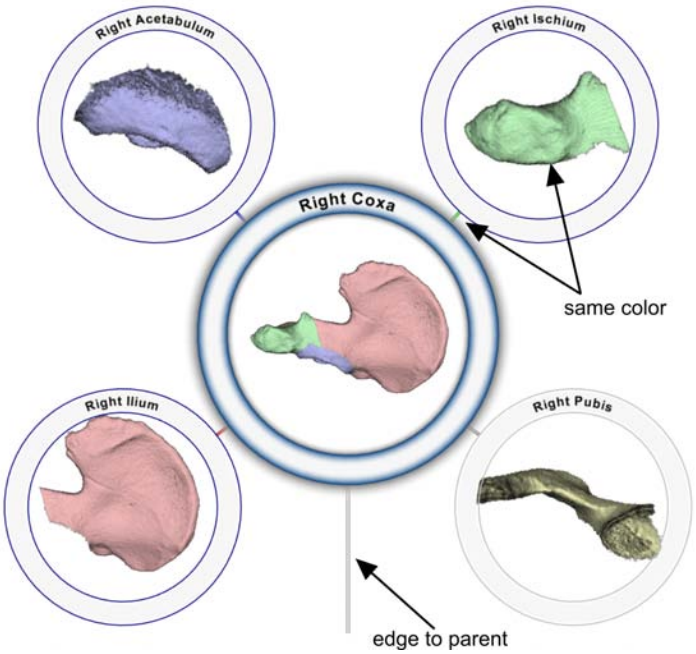
Figure 3.4: A subset of a hierarchy showing the cervical curve and its location relative to the skull.

shown with an outline to indicate the location of the cervical curve beneath the skin.

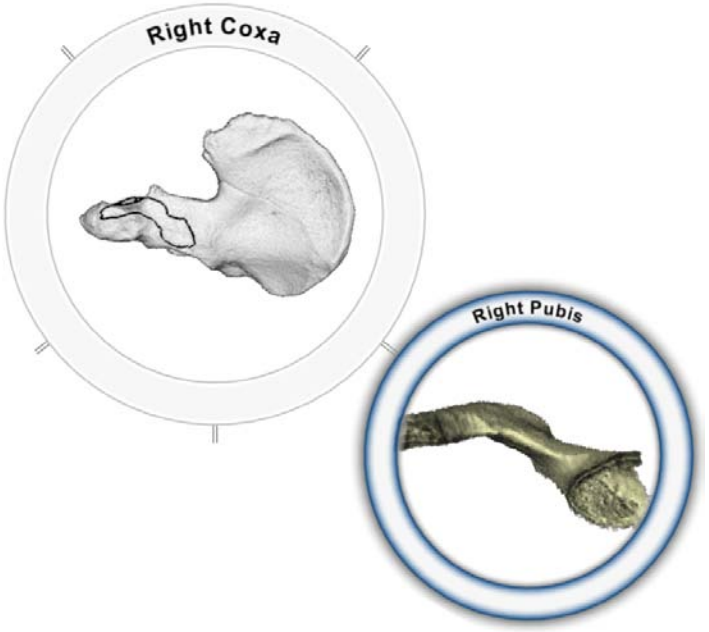
This visualization performs integration at all stages in the pipeline. The volume data is segmented with a labeled hierarchy linked to the segmentations. For every hierarchical node its label and all associated voxels are available. Most of the visualization techniques presented in the two papers use this integrated data-set at all times. Several visualizations integrate at the render stage and additionally produce their own layers. The main layers are the layer produced by the volume-renderer and the node-tree layer containing node circles and connection lines. The outline effect shown in Figure 3.4 is created by utilizing extra calculations performed by the volume renderer. The resulting layer is composited on top of the volume-renderer layer. The final step in the integration is the compositing of the main layers. Care has been taken so that volume renderings are constricted to the available space in a node circle to avoid occlusions between visualizations.

Another visualization that is integrated at the render stage is the occlusion indicator for substructures of a selected object. This can be seen In Figure 3.5(a) where the Coxa has been rotated in such a way that the Pubis is not visible. To indicate that the Pubis is occluded the connection line and the node outline of the Pubis node is colored in gray, where the default color is dark blue. The process that finds the structure boundaries for the outline visualization is also used to count the number of visible pixels for every structure. If the count is below a specified threshold the structure is considered to be fully occluded. This is indicated by looking at the outline created on the Coxa in Figure 3.5(b) where the occluded location of the Pubis is shown.

Studying the MRI slices of the substructures is also a possibility in this visu-



(a)



(b)

Figure 3.5: (a) A substructure of the Coxa, the Pubis, is occluded from the current viewpoint. This is indicated by changing the node-link and node-outline color to gray. (b) The proper location of the Pubis is highlighted with an outline in the Coxa.

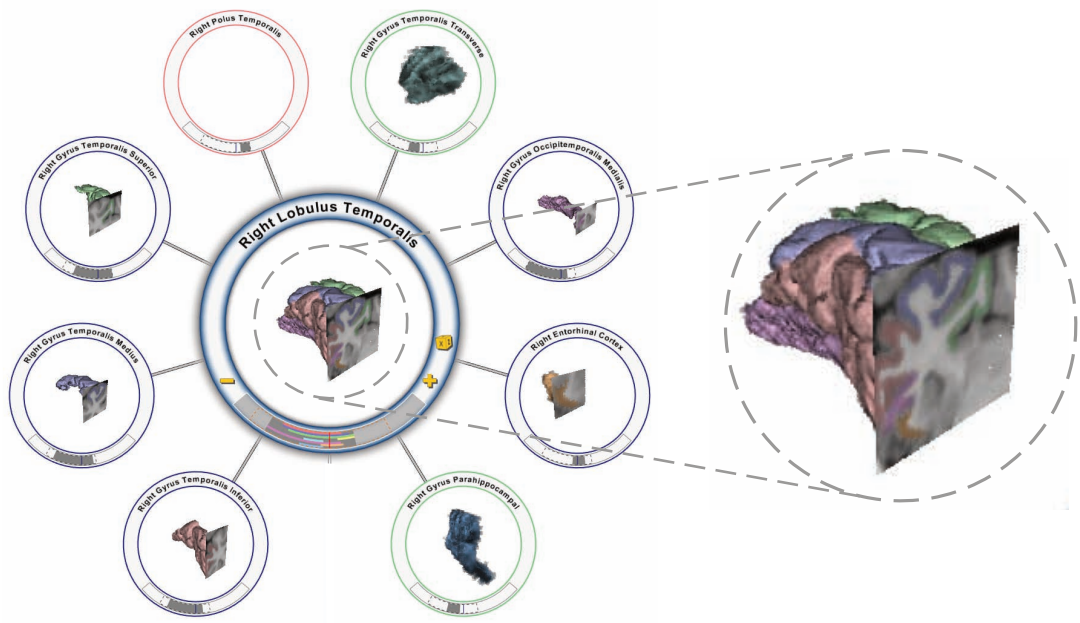


Figure 3.6: Slicing integrated with volume renderering and graph drawing. Node-outline color indicates relative slice position.

alization setup. Figure 3.6 shows a node with slicing integrated into the volume rendering of the temporal lobe of the brain. Since the data-structure is hierarchical and segmented, additional visualization approaches have been implemented. First, every substructure is given its own color so identifying the spatial location of a substructure in the volume rendering or in the slice is simple. Second, the node outline is colored based on the slice position with respect to the structure. The node outline is blue if the structure intersects the slice, green if the slice is before the structure and red if the slice is behind the structure. Third, at the bottom of the node circle a LiftChart like visualization has been added that conveys hierarchical information in addition to segmentation extents.

Together all of these visualizations present a lot of data from different domains in the same view. None of the visualizations occlude each other and the amount of information overload has been reduced considerably. An important aspect of this integrated visualization is zooming in and out from overview to detail. Some of the techniques provide information that makes most sense from an overview perspective such as the node colors used for the relative slice position. Other techniques such as the LiftChart are only legible while looking at the details of a few nodes. The resulting integrated visualization provides a simple user interface to view and interact with a complex data-set.

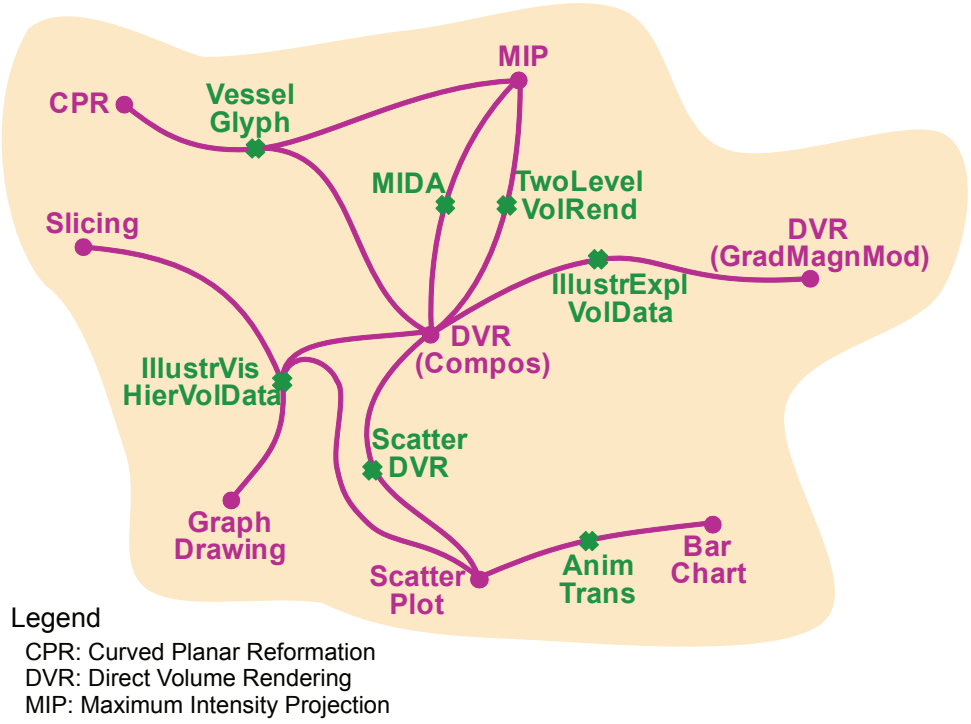


Figure 3.7: $\mathcal{A}$-space with example population.

3.4 $\mathcal{A}$-Space

We have investigated many different types of data in the course of this work and we have employed many different visualization techniques. It is our opinion that with increased data complexity the need for integrated visualizations will also increase. $\mathcal{A}$-space is a more systematic way of looking at visualization integration and moves visualization research away from creating integrated visualizations in an ad hoc way. $\mathcal{A}$-space is not a space in strict mathematical terms, but may be a useful tool for classifying and indicating the possibilities of integrated visualizations. In $\mathcal{A}$-space visualization algorithms are considered to be points and creating an integrated visualization that is based on two or more existing points is considered as *interpolation* or *reconstruction* in $\mathcal{A}$-space. Both of these operations can be interpreted as *blendings of algorithms*.

Figure 3.7 shows a sketch of what $\mathcal{A}$-space may look like. The pink points represent known algorithms that in principle are not integrated visualizations. Between these points paths have been drawn with green crosses indicating the reconstructed algorithms. MIDA is, for example, located in $\mathcal{A}$-space as a blending between MIP and DVR. The work presented in Section 3.3 is also located in the $\mathcal{A}$-space map labeled as *IllustrVisHierVolData*. This work is considered as a blending between DVR, slicing, graph drawing and scatter plots.

Chapter 4

Conclusions

In this thesis we have investigated multi-aspect visualization. In this process we have described two members of this visualization approach, i.e., linked views and integrated views. We have discussed their merits and drawbacks through several examples. We have also presented a concept to create integrated visualizations in the context of the visualization pipeline. The approach addresses the two most prominent challenges in this process, occlusion and information overload. Finally we presented results from our work that are relevant to this topic showing how several aspects of the data can be included in the same view.

It is our opinion that an integrated visualization can be much better than the equivalent linked-view visualization. An argument for this is focus of attention. Since the user's focus is restricted to one view, changes due to interaction are easier to follow. But this typically depends on the application scenario. There are scenarios where having all the information presented separately and unmodified is important. An example of this could be medical workstations where slices in all three slicing directions must be presented as-is and without any distortions due to perspective-foreshortening. In this case a linked view makes more sense than an integrated view.

We have discussed that it is not always easy to create an integrated visualization. Future advances in visualization research may lead to solutions that will make the coupling between volume data and statistical data simpler which should result in easier integration. Currently there is no general solution for this type of problem. $\mathcal{A}$-space may be a possible approach to this. Categorizing visualization algorithms systematically in $\mathcal{A}$may help in identifying possible patterns that will support in creating integrated visualizations.

Since graphics processing power is increasing, the number and complexity of visualizations will increase as well. Coupled with higher resolution displays the number of pixels available for a visualization also increases. These two factors combined means that integrated visualizations can be done at a greater scale than is possible with linked views. If a visualization application needs eight different

views then on average the application only has one eighth of the available pixels for each of the eight visualizations. With an integrated view, all of those pixels are available for a single image. Even while the display resolution increases it still has eight times more pixels than the linked-views.

As the amount and complexity of available data increases, further research into multi-aspect visualization is necessary to provide the visualization techniques that are able to handle this data. We believe that integrated visualizations are very important in this respect. The ad hoc approaches used currently to create integrated visualizations will hardly be sufficient, so more systematic concepts, such as $\mathcal{A}$-space, are necessary.

Bibliography

[1] L. Andersen, S. Berg, O. Gammelsæter, and E. Lunde. New scientific multibeam systems (me70 and ms70) for fishery research applications. *Journal of the Acoustical Society of America*, 120(5):3017, 2006.

[2] S. Bruckner and M. E. Gröller. Volumeshop: An interactive system for direct volume illustration. In C. T. Silva, E. Gröller, and H. Rushmeier, editors, *Proceedings of IEEE Visualization*, pages 671–678, Oct. 2005.

[3] S. Bruckner and M. E. Gröller. Style transfer functions for illustrative volume rendering. *Computer Graphics Forum*, 26:715–724, 2007.

[4] S. Bruckner and M. E. Gröller. Instant volume visualization using maximum intensity difference accumulation. *Computer Graphics Forum (Proceedings of EuroVis 2009)*, 28(3):775–782, 2009.

[5] R. Bürger and H. Hauser. Visualization of multi-variate scientific data. In *EuroGraphics 2007 State of the Art Reports (STARs)*, pages 117–134, 2007.

[6] M. Burns, M. Haidacher, W. Wein, I. Viola, and E. Gröller. Feature emphasis and contextual cutaways for multimodal medical visualization. In *Proceedings of Eurographics / IEEE VGTC Symposium on Visualization (EuroVis 2007)*, pages 275–282, May 2007.

[7] H. Doleisch. Simvis: interactive visual analysis of large and time-dependent 3d simulation data. In *WSC '07: Proceedings of the 39th conference on Winter simulation*, pages 712–720, Piscataway, NJ, USA, 2007. IEEE Press.

[8] H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *VISSYM '03: Proceedings of the symposium on Data visualisation 2003*, pages 239–248, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[9] K. Engel, M. Hadwiger, J. M. Kniss, C. Rezk-Salama, and D. Weiskopf. *Real-time Volume Graphics*. A. K. Peters, 2006.

[10] Google maps. http://maps.google.com/, 2009.

[11] D. L. Gresh, B. E. Rogowitz, R. L. Winslow, D. F. Scollan, and C. K. Yung. Weave: a system for visually linking 3-d and statistical visualizations, applied to cardiac simulation and measurement data. In *VIS '00: Proceedings of the conference on Visualization '00*, pages 489–492, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[12] H. Hauser. Generalizing focus+context visualization. In G.-P. Bonneau, T. Ertl, and G. M. Nielson, editors, *Scientific Visualization: The Visual Extraction of Knowledge from Data*, pages 305–327. Springer Berlin Heidelberg, 2006.

[13] Intellicast. http://www.intellicast.com, 2009.

[14] P. Kohlmann. *LiveSync: Smart Linking of 2D and 3D Views in Medical Applications*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, 2009.

[15] J. Krüger and R. Westermann. Acceleration techniques for GPU-based volume rendering. In *Proceedings of IEEE Visualization'03*, pages 287–292, 2003.

[16] S. Roettger, S. Guthe, D. Weiskopf, and T. Ertl. Smart hardware-accelerated volume rendering. In *Proceedings of the Symposium on Data Visualisation VisSym'03*, pages 231–238, 2003.

[17] Simvis. http://www.simvis.at, 2009.

[18] M. Termeer. *Comprehensive Visualization of Cardiac MRI Data*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, 2009.

[19] C. Tietjen, B. Meyer, S. Schlechtweg, B. Preim, I. Hertel, and G. Strauß. Enhancing Slice-based Visualizations of Medical Volume Data. In *Eurographics / IEEE-VGTC Symposium on Visualization 2006*, pages 123–130. Eurographics, 2006.

[20] J. Woodring and H.-W. Shen. Chronovolumes: a direct rendering technique for visualizing time-varying data. In *Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics*, pages 27–34, 2003.

Part II

Papers

Paper I

Sonar Explorer: A New Tool for Visualization of Fish Schools from 3D Sonar Data*

Jean-Paul Balabanian[†] Ivan Viola[†] Egil Ona[‡]
Ruben Patel[‡] Eduard Gröller[†§]

Abstract

We present a novel framework for analysis and visualization of fish schools in 3D sonar surveys. The 3D sonar technology is new and there have not been applications to visualize the data in 3D. We have created an application called Sonar Explorer that satisfies the requirements of domain scientists. Sonar Explorer provides easy and intuitive semi-automatic fish school tracking and survey map generation. The overall pipeline is described and all pipeline stages relevant for visualization are highlighted. We present techniques to deal with 3D sonar data specifics: highly anisotropic volume data aligned on a curvilinear grid. Domain scientists provide initial impressions on interaction and outlook.

1 Introduction

Accurate estimates of fish stocks are necessary for stock assessment and a sustainable fishery. With proper monitoring, the risk for over-fishing and potential recruitment failure is reduced. Modern assessment methods need data on present stock level and distribution, which calls for new *surveillance* technology. In a modern, new ecosystem approach, studies of fish behavior and relations between animals in the water column might be studied by exploiting new technology. Similarly, the fishing industry needs to fish their quota correctly, targeting exactly the size and school volumes they can cope with during the catch process, without harming unwanted species and size groups. New sea surveillance technology may therefore also help sustainable harvesting of the stocks. Advances in underwater acoustic methods using scanning sonar seems to be a promising alternative,

*This work appeared in the proceedings of EuroVis2007.
[†]Department of Informatics, University of Bergen, Norway.
[‡]Institute of Marine Research, Norway
[§]Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria

where now sequences of 3D information is available for visualization and measurement at realistic ranges to cover entire fish schools. Since the systems are also delivering data with output, i.e. calibrated amplitude, they can deliver fairly accurate measures of biomass. The new multi-beam sonar MS70 is a horizontally

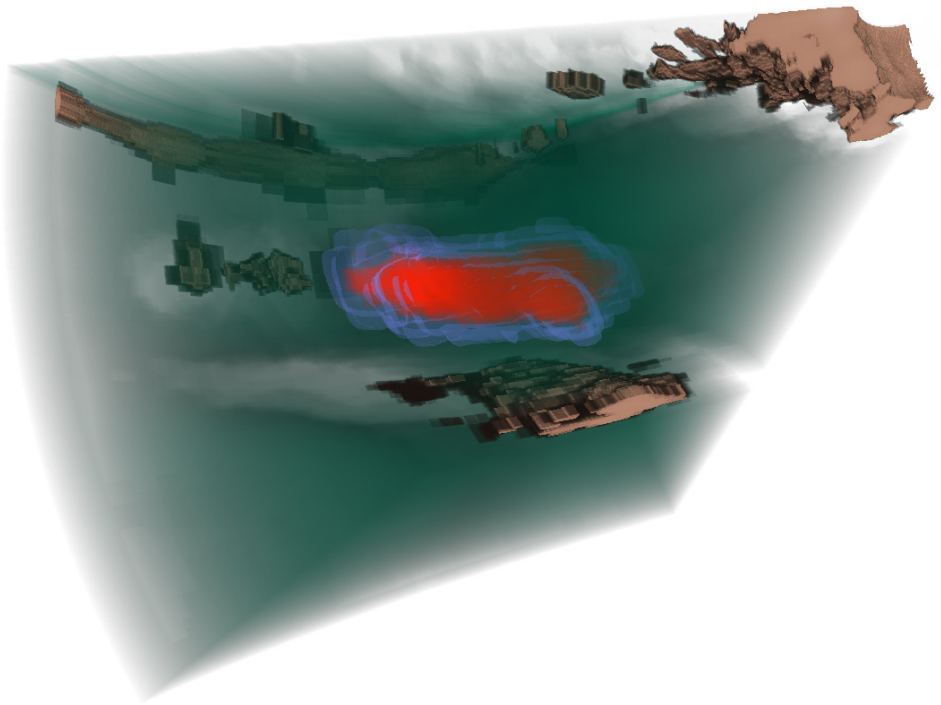


Figure 1: Volume rendering of 3D sonar data.

observing sonar. It yields very high spatial resolution when operating all 500 beams, covering the frequency band 75-112 kHz [1]. The sonar has undergone sea trials from research vessel "G. O. Sars" from December 2005 to December 2006, including detailed calibrations [9]. One of the most challenging tasks, however, is to capture, process and display the data collected by the 800-transducer elements in real time. In standard survey mode, the receiver boards may deliver data at a rate of 2GB per ping (i.e., a one time-step three-dimensional dataset), with a ping rate of 1Hz. Nevertheless, these rates would only be stored in short sequences for research purposes. Normally, these data are transmitted internally within the sonar system to six front-end computers for beamforming and data reduction. Typical data rates delivered to the operator station of the sonar to be stored are 1-2 GB/hour, depending on the range sampled and the sonar pulse repetition frequency (PRF).

At the operator station, two selectable cross sections through the water column can be presented in real time. One cross section is a horizontal slice from a *25-*

beam-fan which shows data from the transducer face to a maximum range of 500 m. The corresponding vertical slice is from a *20-beam-fan* which covers 45 degrees from the surface downwards. Position data from DGPS and accurate motion sensors are delivered to the receiver unit for automatic compensation for vessel roll and heave. The digital range resolution delivered is dependent on the pulse duration used. It is typically 0.38m for the most often used pulse duration of 2ms. As the sonar scans the water column, the data matrix may be used to reconstruct entire fish schools for one ping in 3D. Alternatively if successive pings cover the same school a reconstruction over time in 4D is possible [1].

Efficient surveying, however, necessitates that derivates of the data are analyzed and displayed in near-real time. As realized in the Sonar Explorer, several tasks are readily solved simultaneously, such as a true geographical representation of the vessel, and an effective observation of the detected schools.

The contribution of our paper is the introduction of a framework for visual analysis of fish schools in 3D fishery survey data. This framework satisfies specific needs of domain scientists and provides mechanisms for semi-automatic survey reporting. We provide background information on the data characteristics, present the visual analysis pipeline, and describe how existing visualization methods have to be altered in order to handle specific properties of 3D fishery survey data, i.e.,:

- curvilinear adaption of the GPU ray-caster
- addressing scheme for anisotropic volumes to overcome graphics hardware limitations
- tracking of fish schools in the temporal domain

The paper is organized as follows: Section 2 describes previous work related to our framework. Section 3 describes the individual steps of the pipeline that generates the 3D output. Section 4 highlights pipeline steps relevant to visual analysis and interaction and provides technical details of these steps. Section 5 describe the resulting survey map used for fish school distribution analysis. An outlook to future possibilities provided by domain scientists are discussed in Section 6. Finally we draw conclusions and summarize the paper in Section 7.

2 Related work

Our work intends to assist fishery industry and marine research in estimating stocks for sustainable fishery [5]. The estimation is based on measurements by different acoustic scanning devices [8]. Sonar Explorer performs visual analysis on 3D surveys obtained by MS70 where the measurements are taken over several hundred time-steps. Software provided by the hardware vendor [12] has very

limited functionality allowing only slicing of the curvilinear dataset in two orthogonal directions. There have been only very few attempts to visualize features of 3D surveys until now. Applied methods have been 3D volume rendering and multi-planar reconstruction of time-varying data [1, 9]. The results did not offer any interaction possibilities, as these precomputed animations were intended for presentation purposes. In contrast to these early results, the functionality of the Sonar Explorer is intended for visual analysis and feature (fish school) identification.

One time-step of the entire survey, denoted as a ping, is a volumetric dataset aligned on a curvilinear lattice. Direct volume rendering of 3D curvilinear grid data has been the subject of research for many years [4, 13]. Our curvilinear volume data corresponds to a *conical* cutout of a sphere, therefore the rendering is significantly simplified as opposed to handling general curvilinear grids. This property allows us to use a modified version of a standard GPU ray-caster [10, 7] with addressing using spherical coordinates.

Our framework performs visual analysis of three-dimensional time-varying scalar data. Frameworks for visual analysis of data with similar characteristics such as SimVis [2, 3], provide, up to some extent, similar functionality. The difference is driven by a special handling of the underlying data as they originate from different science domains.

Our aim was to present the results of the visual analysis as clear as possible. This aim was achieved by incorporating focus+context methods [6] where the goal is the possibility to analyze individual pings and see the distribution of all features in a linked survey map.

3 Overall Processing Pipeline

The 3D sonar MS70 is the first device that allows three-dimensional screenings of sea resources over time. The overall goal is to study and precisely quantify fish resources. Also the three-dimensional distribution of fish schools in the sea according to properties such as sea temperature or season is of high interest. The entire pipeline starts with the **data acquisition**. The research vessel performs 3D sonar measurements co-registered with DGPS position and UTC time in the scope of a *survey*. A survey consists of several 3D measurements of interesting areas over time. These measurements are denoted as *observations* and each observation consists of a set of single time-step 3D datasets denoted as a *ping*.

The **Sonar Explorer** allows to **visually analyze** entire survey data in order to semi-automatically generate survey maps of a sea region where the scanned fish schools are clearly depicted. This is an iterative process. The user analyzes every observation of the survey individually. For each observation all time-steps are explored. If a school is present, we provide a robust and easy way to **select**

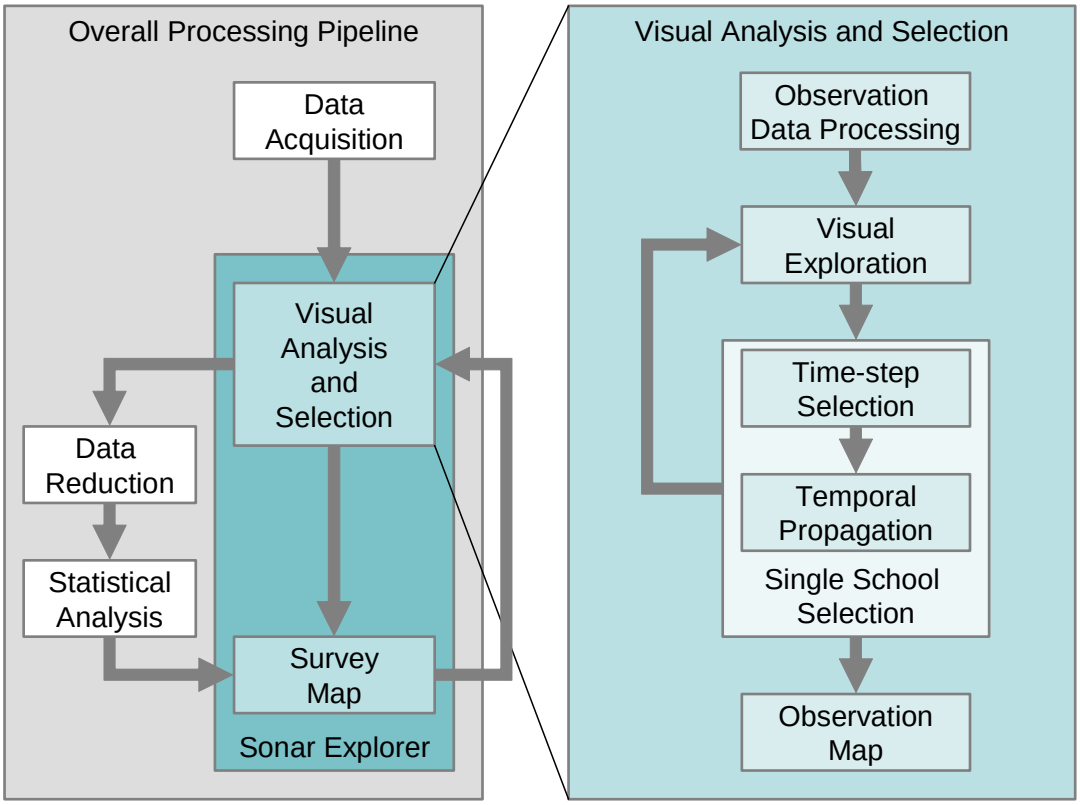


Figure 2: The overall processing pipeline with the Sonar Explorer pipeline in detail.

the school volume. After all fish schools of an observation are selected, this information is exported to the **survey map** and the visual analysis of the next observation is performed. Processing all observations results in a survey map of all identified fish schools.

Our system supports export of the segmented features from our application to allow processing beyond visualization. Exporting only selections can significantly help to **reduce the data** size of the entire survey. After storing the relevant data only, noise artifacts, entire pings or even all pings of an observation can be removed from the survey. In case when there is no interesting data in an observation, only the vessel path with time-stamps will be stored. The export is also important for further information extraction from the school data. The reduced size of the survey data enables processing of the data by **statistical analysis** packages to retrieve more information about the survey. Such information can be then included into the generated survey map. The overall processing pipeline is depicted in Figure 2.

4 Sonar Explorer

Sonar Explorer provides tools that enable a user to visualize 4D sonar data. The visualization tools aid the user in looking for fish schools and selecting these for immediate overview and later analysis. The right part of Figure 2 shows the pipeline architecture followed in the application.

The acquired data for a single observation is first loaded into memory, processed for noise, and localized according to DGPS coordinates. This is performed in the **observation data processing** stage. After being processed, the data is available for **visual exploration**. It is at this stage that the user can search for fish schools. When a fish school has been located it is possible to perform **single school selection**. This process is divided into two parts. The first part is the **manual selection** of a single school in one time-step. The other part is to let this selection automatically be **propagated** in the temporal domain to neighboring time-steps. At the final stage the user can search for other schools or investigate the results in the **observation map**. We now take a closer look at the considerations that have been made during the design of our framework.

4.1 Dataset Characteristics

The data that the application has been designed for is generated by a sonar device. The device takes 3D images at a specified interval while the research vessel is moving. This generates a large 4D dataset. Due to the properties of the sonar device the data is given on a curvilinear grid. More specifically the covered volume is a cone in spherical coordinates. Since many existing volume rendering technologies are designed for volumes given on a rectilinear grid we just had to adapt the standard raycasting technique to our type of grid. A typical hardware accelerated technique uses proxy geometry to find startpoints and endpoints of all visible rays. The usual proxy geometry is a cube. This does not work efficiently with our data so a more suitable proxy geometry was needed. Additionally due to a limitation of current graphics hardware concerning 3D texture sizes, we had to use a more sophisticated storage scheme as a work-around.

The MS70 is a sonar transducer consisting of 25x20 beams. These beams are positioned in a grid and they are pointing in a 60 degree horizontal angle and 45 degree vertical angle. Each horizontal array is called a fan and transmits a signal at a specified frequency and then listens for the reflected signal. The reflected value is measured in decibel (dB). The number of samples along a ray is configurable. Our data has 1319 samples per ray and with a sampling distance of 0.38 m this equals to a beam length of approximately 500 meters. The data file has a ping interval of about 5-6 seconds. In addition to the volume data other devices on the vessel also provide information such as DGPS coordinates, UTC timing, and the dynamics of the vessel including heading, pitch, heave, and

roll. This data is introduced into the data stream at a datagram rate of 10 Hz. Due to the nature of the sonar device, the returned signal strength is reduced with geometrical spreading and absorption. To compensate for this a time varied gain function [11] could be applied. Noise is also an issue with the given data. We remove the fan closest to the surface because this fan contains noise from sea surface waves and air bubbles. We also remove the first 15 meters of slices parallel to the transducer since the noise here is produced by the sonar device itself. Other noise types that occur are: high intensity walls and high intensities along a beam sometimes occurring when loosing data packets in the data stream during the trial survey. These artifacts are strongly suppressed by applying median filters. These four noise types are illustrated in Figure 3.

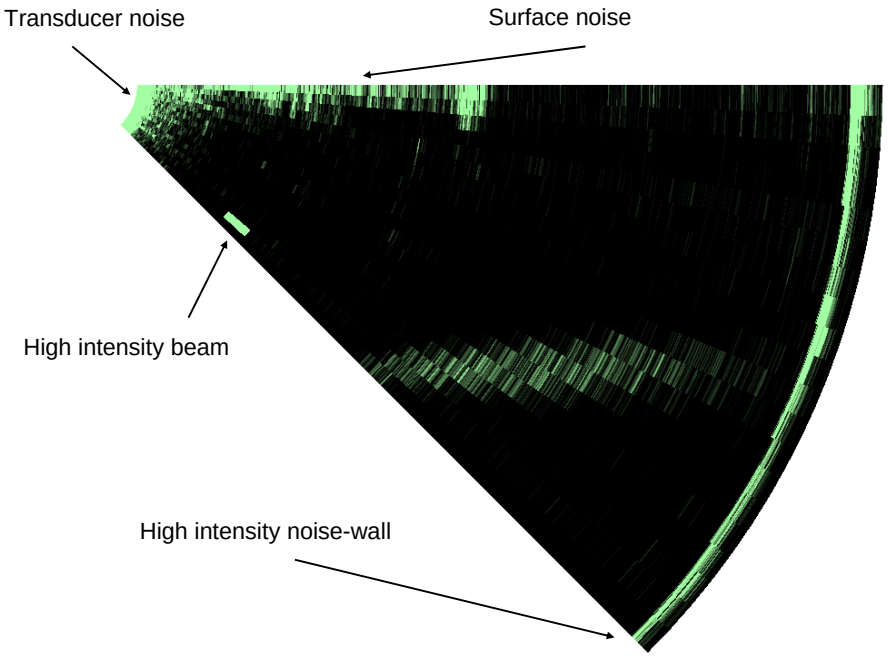


Figure 3: Examples of typical noise in 3D sonar data: transducer noise, water surface noise, high intensity beam, and high intensity noise-wall.

The sonar cone's center is located at the transducer. The curvilinearity of the volume can be represented using spherical coordinates. The following equations describe the conversion from spherical coordinates (θ, ϕ, ρ) to Cartesian coordinates (x, y, z) :

$$\begin{aligned} x &= \rho \sin\phi \cos\theta \\ y &= \rho \sin\phi \sin\theta \end{aligned}$$

$$z = \rho \cos \phi$$

These equations describe the conversion back to spherical coordinates:

$$\begin{aligned} \rho &= \sqrt{x^2 + y^2 + z^2} \\ \theta &= \tan^{-1}\left(\frac{y}{x}\right) \\ \phi &= \cos^{-1}\left(\frac{z}{\rho}\right) \end{aligned}$$

In computer memory the voxel values are positioned on a regular grid. The grid is interpreted as having coordinates in spherical coordinate space. This simplifies the conversion between a position in spherical coordinates and the corresponding position in Cartesian coordinates. The raycaster that we have adapted processes the start and end ray positions in spherical coordinates. To traverse the ray through the volume from the start point to the end point, the position is converted to Cartesian coordinates before calculating the next ray position. After the new position has been located the position is converted back to spherical coordinates before the sample value is retrieved. This is done because a linear ray in Cartesian coordinates corresponds to a curved ray in spherical coordinates.

The volume is highly anisotropic. A technical problem that we encountered is due to limitations of NVidia's implementation of 3D textures. NVidia has limited the dimension of these textures to 512^3. Our volume data contains more than 512 samples in one direction. This problem is solved by folding the volume in the RGBA components. This technique works by putting the first 512 samples in the red component then putting the next 512 values starting with the 512th sample in the green component. This is continued in the same way for the blue and alpha components. With this approach we can have a volume with dimensions of up to 512x512x2045. The reason we do not have 2048 in the last dimension is linear interpolation. To achieve linear interpolation in hardware, we need to repeat the last samples in one fold as the first samples in the next one. A GLSL shader takes care of returning the correct value from the appropriate component during raycasting and texturing. In general this technique can be used to handle anisotropic volumes with different configurations. Figure 4 illustrates two different configurations that might occur and the color component that could be used as a mapping.

4.2 Visual Analysis

The visualization part of the application is the most important interaction tool. It aids the user in finding schools, enables him to extract the features that are interesting, and gives the user a context in which the information makes sense. In

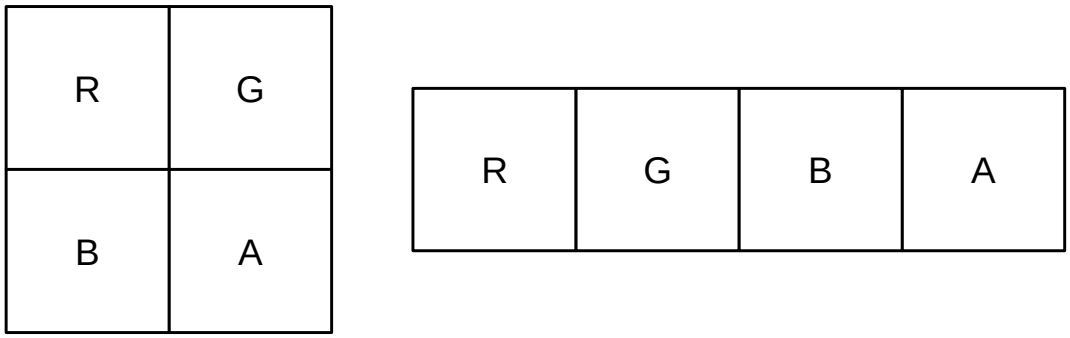


Figure 4: Two possible texture folding configurations. The R, G, B, and A values denote the color and alpha components used for storing.

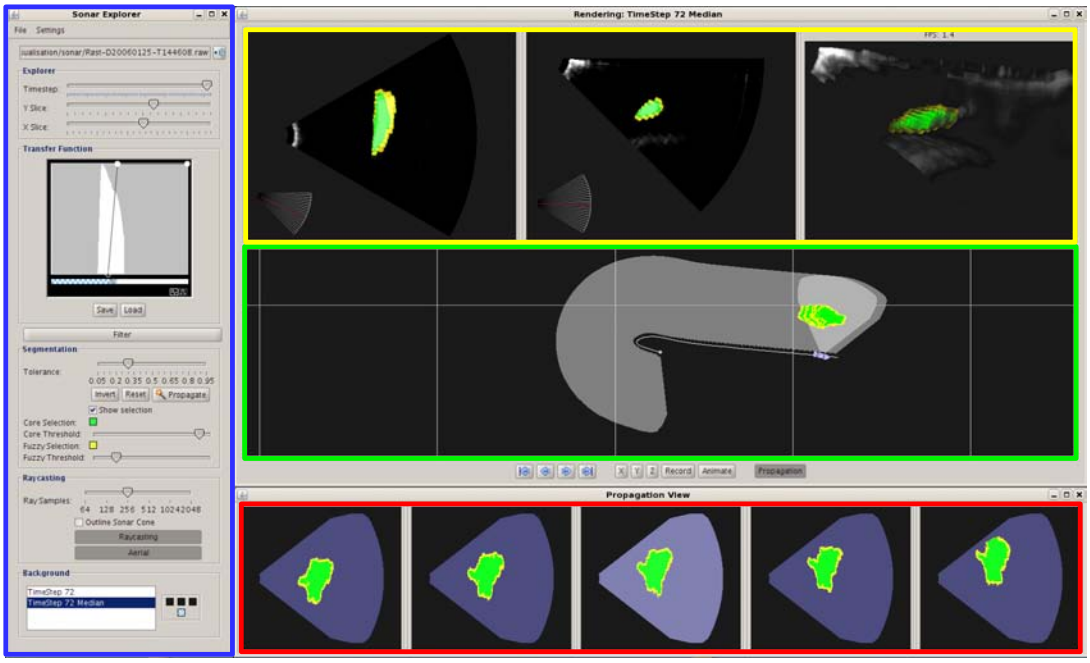


Figure 5: Screenshot of the Sonar Explorer application: Blue: The main control panel of the Sonar Explorer application. Yellow: 2D (top and side) and 3D visualizations of a single ping. Green: Observation map. Aerial overview of observation path (several pings). Red: Propagation view. Orthographic projection of segmentation masks (several pings).

Figure 5 the control panel that provides most of the interaction tools is highlighted in blue. For exploring an observation to find schools we provide 2D slicing and 3D volume rendering. These visualizations correspond to the widgets enclosed by the yellow rectangle in Figure 5. The two viewports on the left show axis aligned

slices from the top and side of the sonar volume. The viewport on the right displays a direct volume rendering of the data. Both the slices and the volume rendering are dependent on the chosen transfer function. To give the user some context we provide an overview. The overview displays the path traveled during acquisition of the observation, where the current volume is located in this path, and a highlighting of any segmented data. The green part of Figure 5 shows the top-down aerial view displaying the overview in the so-called observation map. The application also provides a way of showing the results from neighboring time-steps. The red part of Figure 5 shows the propagation view which displays the current volume as well as the two previous ones and the two following ones.

The 2D slices provide an elementary way of exploring the data. These views are the ones that represent the data with highest accuracy since they are displayed without any filtering. The two viewports to the left in the yellow part of Figure 5 display axis-aligned slices from the top and from the side. Any given transfer function will change the output of this view. The slicing view allows the user to select data that are fish schools. The user utilizes the mouse to click a point in the slice and the coordinate of this point are used as a seed point in a region growing algorithm. After a segmentation mask is created for the selection, it is blended with the displayed data to highlight the selected area.

The right viewport in the yellow part of Figure 5 displays the direct volume rendering of the sonar data. The volume rendering is an OpenGL GPU based raycaster adapted to a curvilinear grid. The algorithm achieves a speedup by rendering a proxy geometry that defines which pixels contain data that need to be raytraced. Default raycasting algorithms use a cube as proxy geometry but we have modeled the sonar cone and are using this as our proxy geometry. Analogous to the 2D slice view, the 3D rendered view is also using a given transfer function configurable from the control panel shown in the blue part of Figure 5. The transfer function defines which densities are transparent and which provide a visible contribution. Figure 1 illustrates the results that are possible by only adjusting the transfer function. Due to the high anisotropy of the volume some rendering artifacts occur. This can be resolved by increasing the number of samples along a ray but penalizes the performance.

The observation map is an aerial overview of the current observation. This view shows the vessel, vessel path, the convex hull of all ping locations, highlighting of the current ping location, and an orthographic projection of the segmented schools. The projection uses the same basic raycasting algorithm supporting curvilinear grids but the projection is now orthographic. The raycasting in this view performs Maximum Intensity Projection (MIP) of the segmentation mask of the current volume.

To let the user see how the fish schools move over time, we provide the propagation view. The propagation view displays the projection of the selection mask of the current volume and the selection masks of adjacent time-steps.

If segmentation data is available the red, green, and yellow parts of Figure 5 will highlight this fact. The segmentation mask contains float values that represent the confidence of the segmented voxels being part of a fish school. In all the view-ports in Figure 5 one can see the highlighting as the green and yellow blobs. The green center is the core area and the outer yellow area is called the fuzzy area. These areas can be changed by modifying two threshold values. To highlight the boundary of the feature in the 3D view, we use first hit rendering to display the fuzzy area. The core area is raycasted using alpha accumulation.

4.3 School Selection

As previously mentioned, the school selection is heavily aided by the visualization. The visualization provides the user with a way of searching for and identifying schools of fish. After segmenting a feature the visualization highlights the selected area and the user can now in 2D and 3D check if there are parts of the school that have not been included in the selection. In addition the 3D view gives a context for the 2D slices. The user can in 3D see where in a volume the school is located and can then navigate the slices to this area. Another way of exploring is by adjusting the transfer function. Adjusting the transfer function will enable the user to a certain degree to highlight interesting values and suppress values that do not contain any useful information. Choosing a good transfer function results in better segmentation since the algorithm adapts to the given transfer function. The selection process is first performed in one time-step and then propagated to neighboring time-steps.

Feature extraction in one time-step is a two step process. First the volume is filtered and then any interesting features are manually segmented using region growing. The filtering is typically performed with a median filter. Due to the anisotropic nature of the volume we also employ an anisotropic 3D kernel. The kernel's dimensions mimic the anisotropy of the volume. We have used a kernel with dimensions 3x3x23. One aspect we do not consider is that the vertical and horizontal distances between voxels changes linearly as the kernel moves towards the perimeter.

After filtering the user will manually select a school through visual identification in the volume. The segmentation is based on a flood-fill algorithm that grows from a user-given seed point. The alpha value from the transfer function for a voxel is used as a basis for the flood-fill algorithm. The absolute difference between the seed point's value and the value of the currently processed voxel is used to find out if a voxel is part of the fish school. A user provided tolerance value decides how large this difference can be before a voxel is discarded. A tolerance slider is visible in the control panel in the blue part of Figure 5. The difference and the tolerance is also used to calculate a mask value. This mask value is one when the difference is zero and zero when the difference equals the

tolerance value. We use a fall-off function to create the values in between. The present function implemented is:

$$f = 1 - \left(\frac{d}{t}\right)^2$$

Where f is the falloff, t is the tolerance, and d is the absolute difference between the seed point value and the currently processed voxel. We always clamp the fall-off value to the range: $[0, 1]$.

After a school has been identified and segmented in one time-step, the user can use the segmented school as a template for propagation to neighboring time-steps. First we determine the current tolerance of the segmented area. Then we calculate the center of gravity of the school and the bounding box. These properties are used to effectively propagate the school both forward and backwards in time.

To be able to have a seed point that will be valid in a neighboring time-step, we calculate the center of gravity with the assumption that the school is convex. The center of gravity that we calculate is weighted with the segmentation mask values. During this calculation we also determine the bounding volume of the mask. The bounding volume is used in the propagation to limit the flood-fill from growing into the sea bottom or up to the surface. We increase the size of the bounding volume to consider that the size of the selection may change from time-step to time-step. Currently this technique will stop the propagation before the school leaves the volume. Segmenting schools that are partly in a volume will be subject of future work.

Since the selected feature usually is the result of multiple flood-fills the selection's actual range of values will exceed the range of the tolerance. So any automated region growing using this tolerance will be sub-optimal. Our solution to this is to find the range of the selection and then calculate the tolerance that will cover this range. We also determine the density that corresponds to the center of the tolerance range. This center value is used in the region growing as a comparison value instead of the value located at the seed point determined by the center of gravity. This makes the tolerance value more effective.

By using the center of gravity from one time-step as a seed point in the next time-step we have an estimate of a location that should still be inside the school. After the region growing is completed, we recalculate the center of gravity and the bounding volume. We increase the size of the bounding volume and if the bounds are outside of the sonar volume we stop the propagation. This means that schools that are only partly in the volume will not be segmented.

5 Results

At the end of our pipeline we have two results. First we provide a way for the user to save the selection masks. The selection masks give the statistical analysis

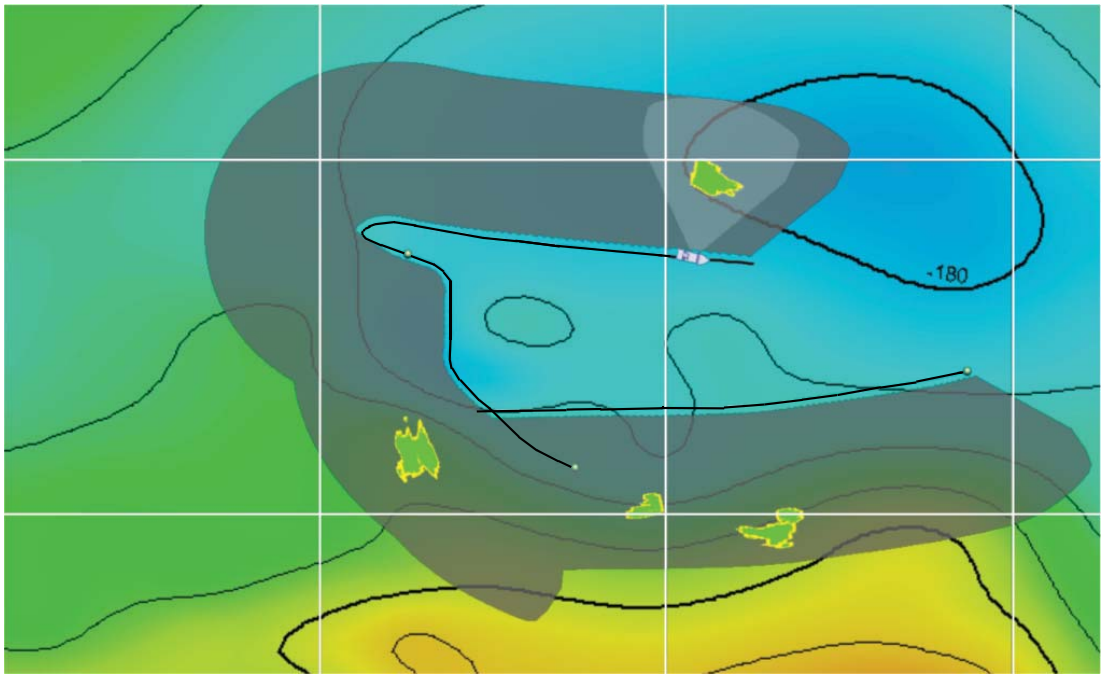


Figure 6: Survey map showing detected herring schools. Rough depth contours are indicated in the background, with the sonar search volume shown in grey.

tool the information it needs to perform calculations that are interesting to the user. The other result from our application is the survey map. The survey map illustrates for the user the survey path but also the location of schools. In Figure 6 the survey map has been manually composited with a colored depth map to provide context for the data. The survey map shows the results from three different observations. Each of the small green spheres indicates an observation starting point. The missing observation between the research vessel and the lower right observation was dropped because it did not contain any schools.

Performance: The application operates at interactive speeds but there are some bottlenecks that reduce the performance of the application. On the visualization side there are two tasks that are the major bottlenecks. Volume rendering and observation view rendering. Both of these visualizations use a raycaster to create the results. The volume renderer is slow because of the need to have a high sampling rate to overcome the anisotropy of the volume. The observation view does not have this issue since it renders in a direction that has a low resolution. The performance loss in this view happens if the number of segmented time-steps increases. Due to these issues the application lets the user turn off the volume renderer and the observation view if they are not needed. There is room for optimization and in the future we plan to solve some of the visualization problems by caching results and rendering at a lower resolution during interaction. Another

performance bottleneck is filtering before selection. Median filtering is a demanding algorithm but is in fact currently the filtering method that has provided the best results. We would like to use the GPU in the future to increase the filter performance.

6 Sonar Explorer Use Scenario

The long-term goal for fishery-survey scientists is absolute abundance estimates of stock size for selected, important components of the ecosystem. Herring is now the largest fish stock on the Northern Atlantic, converting maybe as much as 100 million tons of zooplankton to human food. In ecological terms it is therefore a key component both as plankton feeder and prey for predators like cod and saithe. The Sonar Explorer will be an important tool for improving the accuracy of the biomass estimates of herring, how herring utilize its prey field and for quantifying its distribution within national boundaries. This is important for international negotiations when partitioning the catch quota. The uncertainty in the estimate of abundance is now closely connected with how the fish schools react to the presence of the vessel. The new system will certainly reduce this bias on herring stock estimate and hopefully also on the mackerel and capelin stock estimates. Future integration of data from other available instrumentation in a common display has a great scientific potential.

Sea bottom detection and removal, noise estimation, proper school detection with extraction of relevant morphological and energetic parameters of the schools are natural tasks here. Similarly, averaging and volume echo integration over sailed distance in range bins from the vessel to the maximum observation range can also be solved. These displays and outputs will give the operator and cruise leader invaluable information for changing or adapting the survey strategy during transecting. It will also provide data for proper statistical analysis of transect data in time and space.

The Sonar Explorer may also integrate other instruments operating simultaneously onboard the vessel, and which are for the data sampled from the MS70. These are the data from the low frequency fishery sonar SP72, giving one rough slice backscattering from about 2000 meters around the vessel, the EM 300 bottom mapping system, creating a detailed bottom map from a fan of beams under the vessel, the current velocity as measured from the Acoustic Doppler Current Profiler (ADCP), and the vertically observing echo sounders. Tracking the movement of the schools relative to the water masses may then be extracted from combining the information from several sensors.

Sonar Explorer now covers the important missing link between raw data collection and the postprocessing stage of the data from the sonar. The true geographical representation gives the cruise leader a good overview of the density

distribution and structure, necessary for immediate decisions for adaptive sampling strategies. The possibilities for data reduction and detailed school analysis are important tools in surveys for pelagic fish.

7 Conclusions

In this paper we have presented Sonar Explorer, an application framework for semi-automatic generation of fish school 3D survey maps. The fish school selection is carried out by a visual analysis and and easy and intuitive selection via picking. This selection is automatically propagated in time by tracking fish schools in the temporal domain. We have presented various aspects of the underlying data that imply modifications to the standard GPU based volume rendering.

The 3D survey of fish schools results into a survey map where all identified schools are located. Identification of volumes with and without school data can be used as an efficient data compressor. The survey and the observation map serve as contextual views that are linked to the focus view (2D and 3D view) where individual pings can be analyzed. When used in real time, the system offers the cruise personnel an immediate overview of the survey situation for immediate actions, and provides tools for a detailed analysis of single schools. Further connectors to survey charts and statistical analysis systems are foreseen.

8 Acknowledgments

We thank Rolf Korneliussen for the data, Stefan Bruckner for helping with the GPU raycaster, and Torsten Möller for many interesting discussions.

References

[1] L. Andersen, S. Berg, O. Gammelsæter, and E. Lunde. New scientific multibeam systems (me70 and ms70) for fishery research applications. *Journal of the Acoustical Society of America*, 120(5):3017, 2006.

[2] H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+context visualization of complex simulation. In *Proceedings of VisSym'03*, pages 239–248, 2003.

[3] H. Doleisch, G. Stonawski, and H. Hauser. Simvis: Interactive visual analysis of simulation results. In *Proceedings of the NAFEMS Seminar on Simulation of Complex Flows (CFD)*, 2005.

[4] M. Garrity. Raytracing irregular volume data. In *Proceedings of SIGGRAPH '90*, pages 35–40, 1990.

[5] D. Gunderson. *Surveys of Fish Resources*. John Wiley & Sons, 1993.

[6] H. Hauser. *Scientific Visualization: The Visual Extraction of Knowledge from Data*, chapter Generalizing Focus+Context Visualization, pages 305–327. Springer, 2005.

[7] J. Krüger and R. Westermann. Acceleration techniques for GPU-based volume rendering. In *Proceedings of IEEE Visualization '03*, pages 287–292, 2003.

[8] D. MacLennan and E. Simmonds. *Fisheries Acoustics*. Chapman & Hall, 1993.

[9] E. Ona, J. Dalen, H. Knudsen, R. Patel, L. Andersen, and S. Berg. First data from sea trials with the new ms70 multibeam sonar. *Journal of the Acoustical Society of America*, 120(5):3017, 2006.

[10] S. Roettger, S. Guthe, D. Weiskopf, T. Ertl, and W. Strasser. Smart hardware-accelerated volume rendering. In *Proceedings of VisSym '03*, pages 231–238, 2003.

[11] SIMRAD. Simrad ek500. theory of operation. Technical report, Simrad, Norway, 1996.

[12] Simrad web site http://www.simrad.com/, 2006.

[13] J. Wihelms, J. Challinger, N. Alper, S. Ramamoorthy, and A. Vaziri. Direct volume rendering of curvilinear volumes. In *Proceedings of SIGGRAPH '90*, pages 41–47, 1990.

Paper II

Temporal Styles for Time-Varying Volume Data*

Jean-Paul Balabanian[†] Ivan Viola[†] Torsten Möller[‡]
Eduard Gröller[†§]

Abstract

This paper introduces interaction mechanisms for conveying temporal characteristics of time-varying volume data based on temporal styles. We demonstrate the flexibility of the new concept through different temporal style transfer function types and we define a set of temporal compositors as operators on them. The data is rendered by a multi-volume GPU raycaster that does not require any grid alignment over the individual time-steps of our data nor a rectilinear grid structure. The paper presents the applicability of the new concept on different data sets from partial to full voxel alignment with rectilinear and curvilinear grid layout.

1 Introduction

Many areas of science, industry, and medicine are nowadays increasingly using time-varying volumetric data sets in their daily routine. Such data sets are usually discretized forms of real-world measurements or results of physical simulations. The nature and usage of time-varying data strongly depends on the given application domain.

A typical application area where time-varying data sets are studied on a daily basis is meteorology. Such data is usually organized on a 3D regular lattice with dozens of characteristic values per sample point. One time-step represents one moment in time and the overall data contains the development over time (e.g., one time-step per hour of a total of 48 time-steps) [12]. Especially for the exploration of hurricane behavior, these simulation results are studied to increase the understanding of how individual properties influence the global behavior of this weather phenomenon.

*This paper appeared in the proceedings of 3DPVT 2008.

[†]Department of Informatics, University of Bergen, Norway.

[‡]Graphics, Usability, and Visualization (GrUVi) Lab, Computing Science Department, Simon Fraser University, Canada

[§]Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria

Probably one of the youngest domains where time-varying data sets have been acquired is marine research. New sea-surveillance technology based on acoustic echo is designed to enable sustainable harvesting of fish stocks and studying fish school behavior [2]. The time-varying data acquired from this technology are partially overlapping pyramids on not aligned curvilinear grids.

The time-varying data sets significantly differ in the number of time-steps from very few to hundreds. The data values might be scalar values or vectors. There might be several data values per single sample, data sets might overlap, or the data values can be organized on a variety of different grids. Despite these differences we can state that effective handling of time-varying volume data is a complex and challenging task. We differentiate between two basic challenges.

The first challenge relates to the computational complexity and memory requirements when processing the time-varying volumes. For example interactive rendering of mid-size volume data organized in a time series is difficult as only few time-steps can be loaded to the dedicated memory of high performance processing units (e.g., GPUs) at once.

The second challenge concerns how to effectively represent a temporal data set visually to enable a particular exploration task. In this paper we aim at addressing this later challenge, in particular how to interact with and how to effectively convey the temporal characteristics of the time-varying volumetric data set.

The focus of this paper is on the easy interaction with visually classified temporal data. The visual classification is carried out through temporal styles that create an intuitive way of condensing information from a series of volumes into a single image. We propose new interaction techniques for selection of visual representations for single image synthesis and animation sequences. Our visualization technique requires to process many volumetric time-steps at once. We propose a multi-volume GPU raycaster utilizing the latest graphics hardware capabilities that can perform ray compositing from multiple time-steps. The multi-volume ray caster features a volume overlap management that handles data with partial to full voxel alignment and defined on rectilinear and curvilinear grid layouts.

Condensing a time-series of bouncing super-ellipsoid data by using temporal styles is indicated in Figure 1. This figure illustrates the difference between rendering of time-steps separately and rendering them as one time space domain in which the image synthesis is carried out.

We provide a brief review on existing visualization approaches for time-varying data in the following Section 2 and a high-level description of our approach in Section 3. In Section 4 we describe the interaction metaphors that we designed for temporal styles and in Section 5 we give a detailed description of our proposed technique. Finally we present our results in Section 6 and conclude with Section 7.

2 Related Work

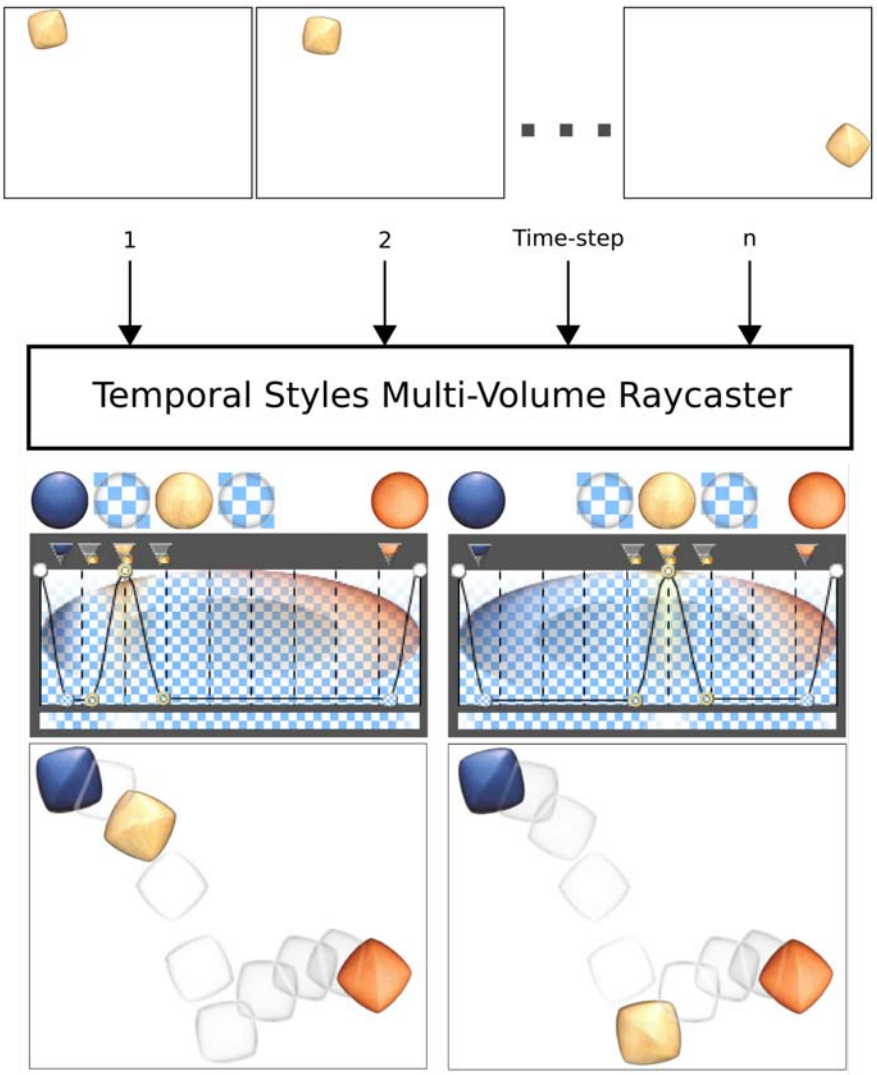


Figure 1: Condensing volumetric time-series using temporal transfer functions. The images show how changing the temporal style generates a new view of the temporal behaviour.

An often used depiction of volumetric data is visualization of a selected single time-step. Such a visualization can be effective to show temporally invariant characteristics. A straightforward approach for visualizing temporal characteristics of time-varying data is time series *playback* as a passive animation or with interactive viewpoint changes. Such visualizations can give a notion of struc-

tured movement, however they will be less useful for most precise analytic and quantitative tasks. In the case when the time series consists of a small number of time-steps, it is possible to use the *fanning in time* approach that shows all time-steps next to each other [8]. However these approaches do not specifically address visual emphasis of temporal characteristics in a time varying data set.

In medical visualization time-varying data has for example been used to plot the temporal development of contrasted blood perfusion as a one dimensional polyline [5, 7]. A similar concept of interactive visual analysis and exploration has been used in data originating from the automotive industry [6]. Other techniques let the user link multi-variate time-varying data with temporal histograms to simplify the exploration and design of traditional transfer functions for this type of data [1]. In the past, techniques have been proposed on automatic generation of transfer-functions by incorporating statistical analysis and coherency [9, 16]. All the above techniques that use automatic or interactive data analysis for exploration and transfer function design employ as visualization single-time-step renderers. This means that information on temporal characteristics is not represented in a single image.

Visualizations mostly related to our approach attempt to visually represent the temporal characteristics of the time-series directly in the physical space. They condense the visual representation so the visual overload is reduced. Some approaches have been inspired by illustration, cartoon or comic drawing techniques where helper graphics like arrows and lines indicate temporal developments such as movement or size changes [10]. The final image consists of several individual time-steps and the helper graphics convey the temporal information. Another approach that we share temporal compositors with, has been inspired by the idea of chronophotography [14]. This technique integrates a time-varying 3D data set into a single 3D volume, called a chronovolume, using different integration functions. When a change of the integration algorithm is requested by the user, the chronovolume has to be recalculated. In contrast to this method the proposed concept of temporal style transfer functions allows interactive visual feedback during the design of the visual representation. Chronovolumes have later been generalized [15]. The technique creates 3D volumes of the time-varying data by slicing the 4D space spanned by the data with hyperplanes and integration operators. The main difference between our rendering technique and theirs, is that we have access to all volumes in real-time. This leads to greater flexibility in interactive design of compositing operators. The visual result of a compositing operator can be easily changed by newly proposed interaction techniques which are the main focus of this paper. The difference between our new and their approach is analogous to the difference between pre- and post classification.

Previous work on visualization of 3D sonar data [3] modifies the standard volume ray-caster to support rendering of parameterized curvilinear volumetric time-steps. The framework gives the user the opportunity to visualize and per-

form semi-automatic segmentation of fish-schools. The segmentation mask from a single time-step is propagated to neighboring time-steps as the initial segmentation mask. It is then automatically adjusted to the new time-step. The temporal aspect of the data, however, can only be visualized in a single time-step at a time.

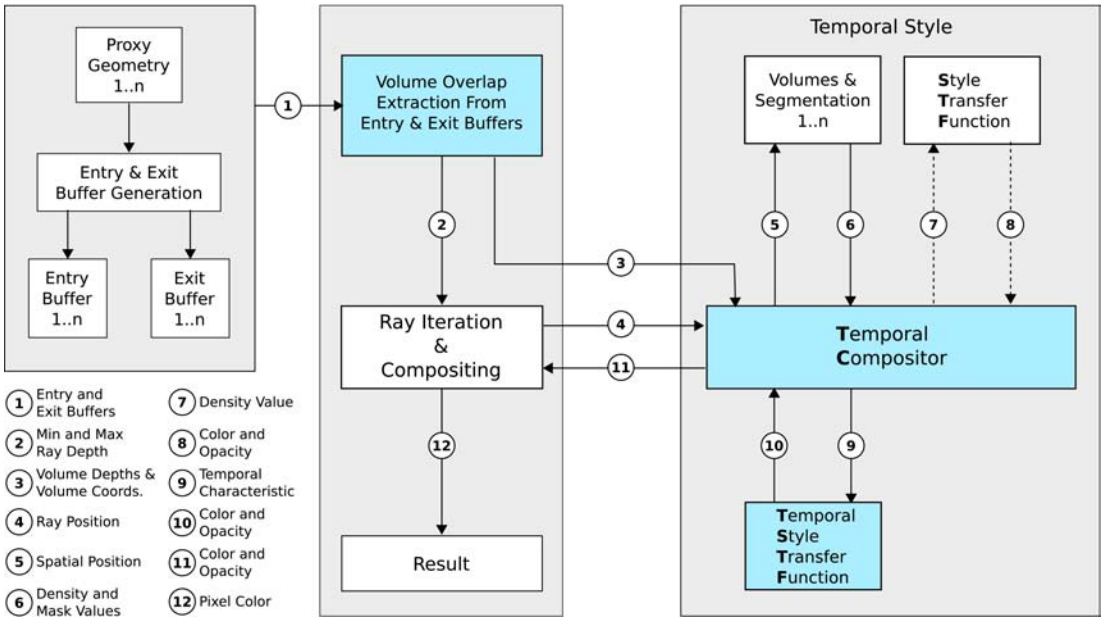


Figure 2: Schematic overview of the TSTF based multi-volume raycaster. Blue indicates our extensions to the standard raycaster. n is the number of volumes. The encircled numbers indicate the processing order. The dashed lines are an optional part of the pipeline.

The concept of style transfer functions [4] describes how images of lit spheres can be used to apply color and texture to volume rendering.

3 Temporal Compositing of Time-Varying Data

The basic idea of temporal style specification stems from the challenge of showing the temporal development of features in time-varying volume data within a single image. Some parallels to this can be drawn from traditional photography where the concept of multiple exposures is well known. The technique creates images that show, for example, where objects move from exposure to exposure in a single image. In Figure 1 we can see the start and stop positions of the bouncing object in addition to the traversed path. In the Figure one can also observe that we are able to change the visual representation of the photographed object, something which normal photography is incapable of doing. Simply by interacting with a widget a user is able to reproduce many of the results generated by long- and

multiple exposure techniques. Generating several images where the exposure changes it is possible to create animations that higlights the change.

Traditional volume raycasting of time-varying data creates images that represent individual time-steps without any temporal information. Even if images from several time-steps are created it is still difficult to compare them spatially and temporally. It is also difficult to identify areas where there is a temporal change. Our aim is to condense several time-steps into one image.

A transfer function is a function that takes density values and converts them to a visual representation. This conversion is usually $R \rightarrow R^4$ and results in a color with opacity.

Style transfer functions [4] are transfer functions that in addition to opacity also define lit-sphere styles for densities instead of colors. Using a density value an opacity and a style is calculated by interpolation. A color with shading is retrieved from a texture of a lit-sphere using additionally the density gradient vector.

The temporal style at a position p is a color and opacity derived from n density values, this can be described as a mapping $R^n \rightarrow R^4$. Our task is to take a density vector in R^n, where n is the number of spatially overlapping voxels from distinct time-steps, and assign a visual representation to it. Our visualization framework calculates values in the temporal domain using a Temporal Compositor (TC) depicted as central blue box in the schematic description of our pipeline in Figure 2. This module processes the spatially overlapping voxels and enables operations on the temporal data. These include temporal average, temporal change, or temporal gradient calculations. The TC takes the n values and converts them to a so-called temporal characteristic ($R^n \rightarrow R$). The result of this conversion using temporal operations can then be applied to a Temporal Style Transfer Function (TSTF), which generates a visual representation $R \rightarrow R^4$.

A temporal style is the visual representation that is generated by the temporal compositor. Depending on the TC the visual representation generated can solely be based on the TSTF or a modulation between the style transfer function and the TSTF. This is usually dependent on the type of information the TC is conveying. In Section 5.3 we describe different TCs that we have implemented.

Our system allows the use of both partially and fully overlapping volumes and volumes defined on regular or curvilinear grids. In Section 5 we give a detailed description of our framework and a detailed explanation of TCs and TSTFs.

4 User Interaction with Temporal Styles

The user has several ways of interacting with the time-varying data. First of all the user can define a time-invariant style transfer function that applies to all volumes and basically defines the visual representation of the content of the

volumes. This is similar to single time-step volume rendering. The user can also interact with a temporal style transfer function which defines the visual representation that the currently selected temporal compositor will use to produce its results. We have implemented two different TSTFs. The first one lets the user supply a modulation for the visual representation for every time-step. The TSTF is divided into sections equal to the number of time-steps. Figure 1 illustrates this metaphore. On the left we have selected a blue style for the first time-step and a light yellow style for the third time-step. In between we have specified a style that only shows the contour. We have also implemented another type where the TSTF represents temporal gradient magnitude and defines a range $[0, 1]$. The value calculated by the TC is then used directly to retrieve a visual representation from the TSTF.

A new interaction type is how the user works with the TSTF. The TSTF contains several styles and nodes that define color and opacity. The styles and nodes can be grouped together and all of the members of the group can be moved simultaneously. This technique is especially applicable for the time-step index TSTF 5.4 where moving a group would be analogous to moving the focus in time. Figure 1 shows this concept on the bouncing super-ellipsoid data set and in Figure 5 the concept has been applied to the hurricane Isabel data set. The styles and nodes that are grouped together are indicated with yellow circles or padlocks.

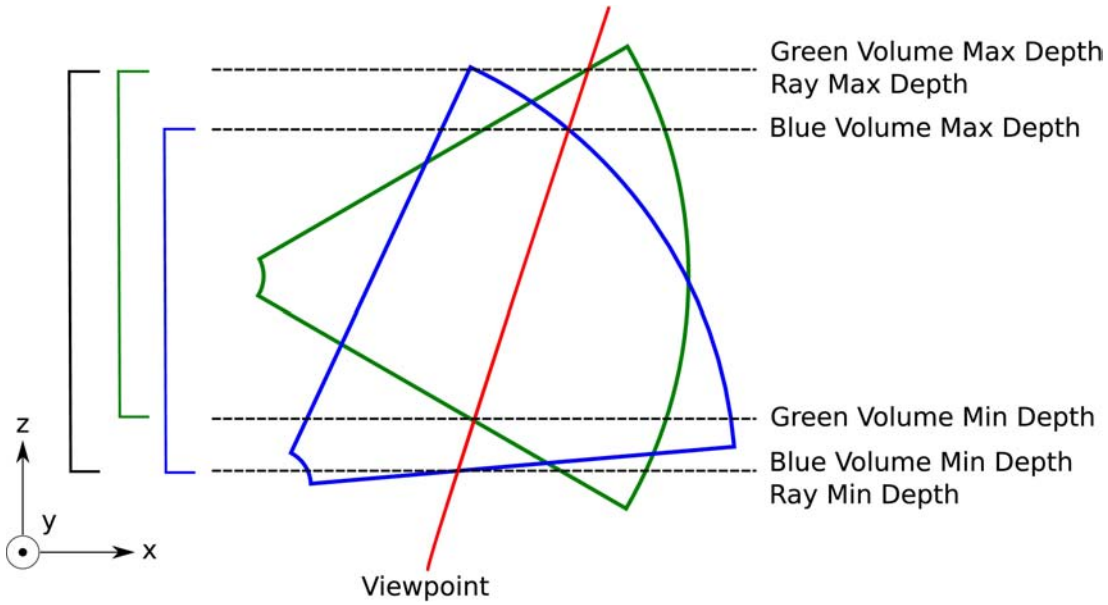


Figure 3: Overview of the minimum and maximum depth values.

5 TSTF Based Multi-Volume Raycaster

The temporal style transfer function based multi-volume raycaster can be realized by extending a standard single volume raycaster. The extensions include the temporal compositor (TC) which handles spatially overlapping voxels and the TSTF that provides a visual representation for the overlap. Figure 2 gives an overview of the system that we have developed. The first step, i.e., the entry and exit buffer generation, is similar to a single volume raycaster, as described by Krüger and Westerman [11]. The difference is that it is repeated for all the volumes that need to be included. To encompass all the steps necessary for multi-volume rendering of time-varying data, various additional steps have been added. First of all, overlapping volumes along a ray have to be identified. In the schematic depiction of our framework (Figure 2) this is done in the module specified by the blue box in the upper left corner. The central issue that has to be addressed is how to represent the temporal information in an intuitive way. In Figure 2 our solution is indicated by the two blue boxes in the lower right corner named Temporal Compositor and Temporal Style Transfer Function. In this part of the process we analyze the temporal behavior and assign a visual representation based on temporal compositing operators and temporal style transfer functions.

5.1 Volume Overlap Extraction

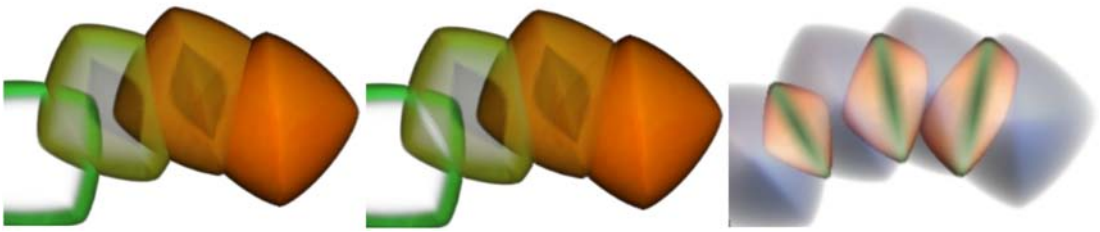


Figure 4: Left: Temporal maximum intensity projection, Center: Temporal average, Right: Temporal maximum density change

To extract the information about volume overlap we generate entry and exit buffer pairs for every time-step. If the position and orientation of the proxy geometry is static for all time-steps only one entry and exit buffer needs to be generated. This process is indicated by the leftmost box in Figure 2. The volume raycasting uses the entry and exit buffer technique for the GPU as previously described [11].

Every pixel corresponds to a single viewing ray. Corresponding pixels in the entry and exit buffers for the various volumes belong to the same ray. This means that the depth information available in the entry and exit buffers describes the amount of overlap for every volume along a ray. By taking the minimum and

maximum depths from all volumes we determine where a ray should start and end. This is illustrated in Figure 3. The process of extracting this information is represented in the pipeline by the blue box in the upper left corner in Figure 2. The Volume Overlap Extraction iterates through every pixel in an entry buffer and checks if the depth is less than the maximum depth of the depth buffer. If this condition holds the volume associated with this entry and exit buffer is intersected with the ray at this position. Every volume is checked and a minimum depth and maximum depth is stored, which is called the ray depth. The ray depth is forwarded to the ray iteration process (path 2 in Figure 2). The volume data coordinates and volume depth values for each volume are forwarded to the temporal compositor (path 3).

5.2 Ray Iteration & Compositing

The ray iteration process takes the ray depth values and, using a suitable step size, samples the ray in regular intervals. For each step the current ray position is forwarded to the Temporal Compositor through path 4 and a temporal style is returned through path 11. The color values are composited, front-to-back along the ray. Using early ray termination, the ray is stopped if the composite opacity is close to 100%.

5.3 Temporal Compositor

The temporal compositor (TC) takes every ray position and checks if it is within the range of the particular volume depth. If the ray position is inside a volume then the sample value, and optionally the segmentation mask value, is included in the temporal compositing (paths 5 and 6). Optionally the TC can also fetch a color and opacity from the time-invariant style transfer function (paths 7 and 8). At this stage the TC has the following values available for multiple volumes: sample value and segmentation mask value, and color and opacity from the style transfer function (STF). The task now is to combine these values in a way that leads to insight into the data. We consider the following strategies:

Temporal maximum intensity projection TC We use the value from the time-step with the highest opacity.

Temporal average TC We average the colors and opacities from all non transparent overlapping time-steps.

Temporal maximum density change TC We calculate the difference between the minimum and maximum density value of all time-steps.

These operators straightforwardly process the underlying data values. However, we support their applicability on pre-classified data by opacity values using the STF as well. This allows for rapid windowing of interesting data values prior to the temporal compositing stage. The opacity defined by the user in the STF

represents visual significance for non-temporal characteristics. It may highlight information that the user is most interested in. The resulting values from these operators, i.e., temporal characteristics, are then applied to a TSTF (paths 9 and 10 in Figure 2).

Figure 4 shows results of using the different TCs on the bouncing super-ellipsoid data set. The image on the left shows the temporal maximum intensity projection TC. The image in the center shows the temporal average TC. The results from these two operators are very similar in this case. The visual difference is that the temporal average has smooth transitions between volumes while the temporal maximum intensity projection has hard edges. The image on the right is rendered with the temporal maximum density change TC and indicates the change from time-step to time-step in the overlapping regions. The green areas are where the change is close to zero while the orange areas depict larger changes.

5.4 Temporal Style Transfer Function

TSTFs are functions that assign a visual representation to results calculated in the TC. We have developed two different TSTFs that correspond to the different outputs generated by the TC. These are:

Time-step index TSTF This TSTF lets the user define time-step visibility and visual representation. The TC provides an index for a single time-step and requests the color and opacity that the TSTF defines for this time-step. Images that use time-step index based TSTF can be seen in Figures 1, 4 (left and center), 5, and 6 (center).

Gradient magnitude TSTF The calculated value received from the TC is the gradient magnitude that is scaled to the range $[0, 1]$. The TSTF represents color and opacity from a style dependent on gradient magnitude. Images that use gradient magnitude based TSTF can be seen in Figures 4 (right) and 6 (left and right).

5.5 Temporal Compositing

The last step for the TC is to take the color and opacity obtained from the TSTF, perform a final processing and then return them to the Ray Iteration & Compositing process (path 11 in Figure 2). The final processing usually consists of deciding whether to return the STF value, the TSTF value or the modulation of those values. If the data contains segmentation information which defines regions of interest the temporal compositing can take this into consideration and only apply the TSTF to the masked region. Then it applies the time-invariant style transfer function for unmasked areas. This technique of highlighting the region of interest has been used on all images showing sonar data in this paper. See Figure 6 (center and right) for examples.

6 Results

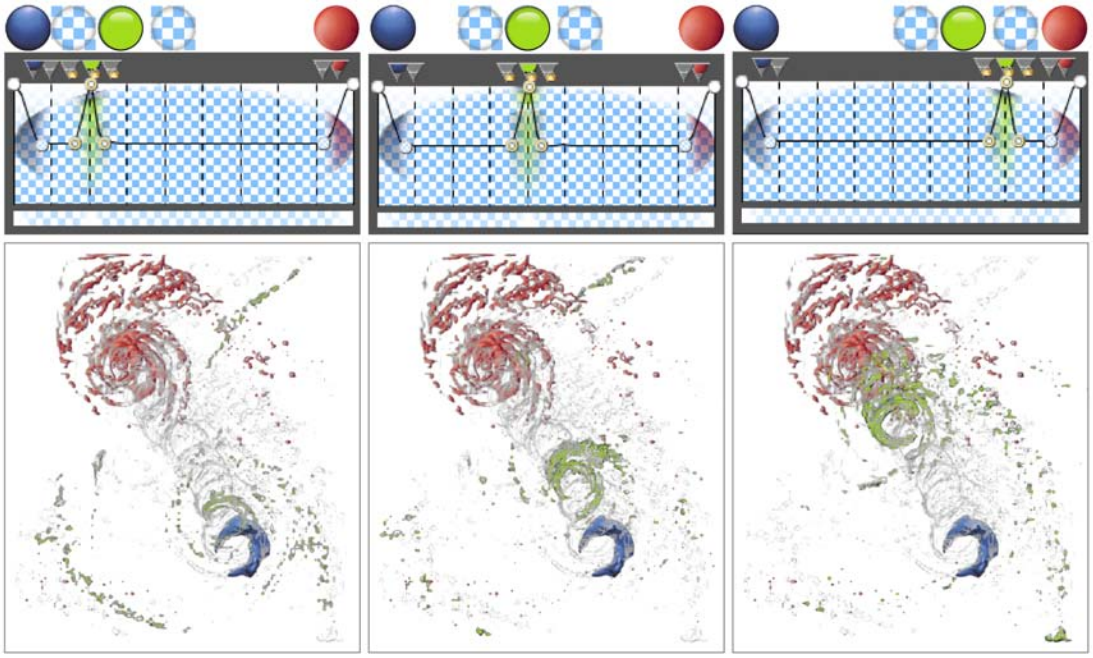


Figure 5: Time-step index TSTF on the hurricane Isabel data set during the interaction. Left: time-step 3 highlighted with a green style, Center: highlight moved to time-step 5, Right: highlight moved to time-step 8.

We have applied our temporal styles concept to three different time-varying data sets. We have applied the temporal average TC and temporal maximum density change TC to the data sets and defined time-invariant style transfer functions and temporal style transfer functions.

The bouncing super-ellipsoid has, throughout the paper, been used to illustrate discussed concepts. The volume data has a resolution of 64^3 and consists of 10 time-steps. The time-steps are partially overlapping and the density values have been calculated from an implicit super-ellipsoid function stored in a rectilinear grid. The bouncing motion has been simulated using a physics library. We also applied the temporal maximum intensity projection TC to this dataset. In the bottom part of Figure 1 we have chosen a blue style for the first time-step and an orange style for the last time-step. In the renderings we can see that the super-ellipsoid at the end-points have fixed positions. We would like to focus on the location of time-step 3. This is achieved by setting the opacity to opaque and a light yellow style for time-step 3. Additionally we set the immediate surroundings of the focused time-step to a low opacity and the silhouette style. The result of this can be seen on the left side as a series of semi transparent objects and an opaque super-ellipsoid at time-step 3. Grouping together nodes and styles of

the focused time-step and moving the group to time-step 6 changes the resulting image. Now the focused super-ellipsoid is located at time-step 6 and the semi transparent super-ellipsoids create paths backwards and forwards in time.

We have also applied the temporal maximum density change TC to the bouncing super-ellipsoid data set (Figure 4). Regions with a low density change have been assigned a green style that turns into an orange style when changes increase. Parts of the super-ellipsoids that do not overlap do not have a defined change and we apply the time-invariant STF.

The hurricane Isabel is a fully overlapping rectilinear data set. We have resampled the *QCLOUD* volume data attribute to the resolution of $125 \times 125 \times 25$ and selected 10 of the original 48 time-steps uniformly distributed over the time-series. In Figure 5 we have applied similar temporal visual settings as to the bouncing super-ellipsoid data set. In the left image we can see where the hurricane starts (the blue region) and where it ends (the red region). It is also easy to see the path of the hurricane as a suppressed contour rendering is set for all time steps and locations of the hurricane. The lower left part of the left image shows the propagating hurricane front for the current time-step (in green). In the center image we have moved the focus to 20 hours from the beginning of the simulated sequence and in the last image it is at 40 hours.

Applying the temporal maximum density change TC to the hurricane Isabel data set we get the image on the left in Figure 6. From this image we can immediately recognize that the areas of highest change are in the hurricane eye towards the direction of the movement.

The final data set that we have applied our technique on is the sonar data. This data is a partially overlapping curvilinear data set. Each volume has a pyramid shape with a resolution of $25 \times 20 \times 1319$. The sequence consists of 75 time-steps (described by Balabanian et al. [3]). We have selected a sequence of 10 time-steps that contain fish-school data and a per time-step segmentation of that school. In the center image of Figure 6 we have set an orange style for the first time-step and a blue silhouette style for the last time-step.

We have also applied the temporal maximum density change TC to the sonar data. The result of this is the right image in Figure 6. The blue style indicates areas of low change and the red style shows high change areas. The center of the school has the highest change which seems reasonable since the density of a school decreases at the edges.

Table 1 shows the performance of the different TC operators. We rendered to a viewport with dimensions 512×512.

By using 3D textures for entry and exit buffers instead of several 2D textures and merging several volumes into a single 3D texture the number of time-steps that we are able to process is not limited to the number of available texture units on the graphics card. Our limitation is the amount of memory available on the graphics card and the maximum resolution of 3D textures (2048^3 on nVidia 8800

Temporal Compositor	Super-Ellipsoid	Isabel	Sonar
Temporal Maximum Intensity Projection	0.20	0.25	2.5
Temporal Average	0.25	0.30	2.1
Temporal Maximum Density Change	0.25	0.25	2.2

Table 1: Rendering times in seconds for a viewport of dimensions 512×512.

GTS).

7 Summary and Conclusions

We have developed a framework for interactive specification of visual parameters for visualization of time-varying data sets sampled on regular and curvilinear volumetric grids with partial overlap. The visualization is carried-out through condensing several time-steps into a single image. To generate this type of visual depiction we used temporal compositors that created a temporal characteristic for the spatially overlapping voxels. We proposed temporal styles to define the visual representation of the temporal characteristics. The resulting images help the user identifying regions of interest in addition to simplifying the interaction by dividing temporal analysis into two components. First a temporal operator, TC, describes a temporal characteristic. Second a temporal style transfer function, TSTF, is designed that highlights only the regions that interest the user.

It is our experience that designing useful temporal styles is just as important and challenging as designing traditional transfer functions. Therefore it is to be expected that a good visual quality of the resulting images will be achieved only after careful design of both, the traditional transfer function as well as the temporal style transfer function. We experienced that the concept of styles significantly simplifies the specification of a useful visual depiction as compared to the traditional transfer functions concept.

The lit spheres concept [13] and consequently style transfer functions have known limitations for shading. Using several differently shaded style spheres in one scene may easily result in inconsistent illumination if the style spheres do not share the same implicitly defined light position. This obviously holds when applying the styles to the temporal domain, however, we have not experienced any undesired visual artifacts related to this limitation. This indicates the known fact that the human observer is not very sensitive to an inconsistent light setup as long as the shape of structures is clearly discernible.

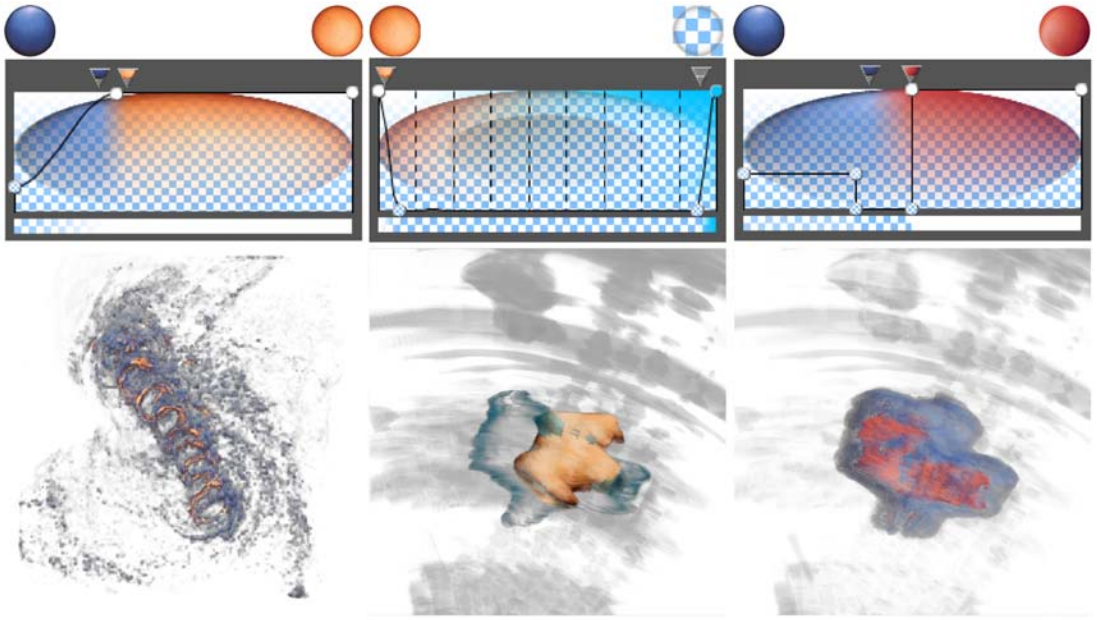


Figure 6: Left: Isabel data set with a maximum density change TSTF, Center: Time-step index TSTF on sonar data, Right: Maximum density change TSTF on sonar data.

Acknowledgments

We thank Daniel Patel for insightful comments and Stefan Bruckner for discussions on style transfer functions. The work has been partially carried out within the exvisation project (FWF grant no. P18322).

References

[1] H. Akiba and K.-L. Ma. A tri-space visualization interface for analyzing time-varying multivariate volume data. In *Proceedings of EuroVis 2007*, pages 115–122, 2007.

[2] L. Andersen, S. Berg, O. Gammelsæter, and E. Lunde. New scientific multibeam systems (me70 and ms70) for fishery research applications. *Journal of the Acoustical Society of America*, 120(5):3017, 2006.

[3] J.-P. Balabanian, I. Viola, E. Ona, R. Patel, and M. E. Gröller. Sonar explorer: A new tool for visualization of fish schools from 3d sonar data. In *Proceedings of EuroVis 2007*, pages 155–162. IEEE, 2007.

[4] S. Bruckner and M. E. Gröller. Style transfer functions for illustrative volume rendering. *Computer Graphics Forum*, 26:715–724, 2007.

[5] E. Coto, S. Grimm, S. Bruckner, M. E. Gröller, A. Kanitsar, and O. Rodriguez. Mammoexplorer: An advanced CAD application for breast DCE-MRI. In *Proceedings of Vision, Modeling, and Visualization'05*, pages 91–98, 2005.

[6] H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+context visualization of complex simulation. In *Proceedings of Eurographics/IEEE-VGTC Symposium on Visualization VisSym*, pages 239–248, 2003.

[7] Z. Fang, T. Möller, G. Hamarneh, and A. Celler. Visualization and exploration of spatio-temporal medical image data sets. In *Graphics Interface 2007*, pages 281–288, 2007.

[8] S. Grimm, S. Bruckner, A. Kanitsar, and E. Gröller. Flexible direct multi-volume rendering in interactive scenes. In *Proceedings of Vision, Modeling, and Visualization'04*, pages 379–386, 2004.

[9] T. Jankun-Kelly and K.-L. Ma. A study of transfer function generation for time-varying volume data. In *Proceedings of Volume Graphics '01*, pages 51–68, 2001.

[10] A. Joshi and P. Rheingans. Illustration-inspired techniques for visualizing time-varying data. In *Proceedings of IEEE Visualization '05*, pages 86–93, 2005.

[11] J. Krüger and R. Westermann. Acceleration techniques for GPU-based volume rendering. In *Proceedings of IEEE Visualization'03*, pages 287–292, 2003.

[12] National center for atmospheric research. http://www.ucar.edu/, 2007.

[13] P.-P. J. Sloan, W. Martin, A. Gooch, and B. Gooch. The lit sphere: a model for capturing NPR shading from art. In *Graphics interface*, pages 143–150, 2001.

[14] J. Woodring and H.-W. Shen. Chronovolumes: a direct rendering technique for visualizing time-varying data. In *Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics*, pages 27–34, 2003.

[15] J. Woodring, C. Wang, and H.-W. Shen. High dimensional direct rendering of time-varying volumetric data. In *Proceedings of IEEE Visualization'03*, pages 417–424, 2003.

[16] H. Younesy, T. Möller, and H. Carr. Visualization of time-varying volumetric data using differential time-histogram table. In *Workshop on Volume Graphics 2005*, pages 21–29, 2005.

Paper III

Hierarchical Volume Visualization of Brain Anatomy*

Jean-Paul Balabanian[†] Martin Ystad[‡] Ivan Viola[†]
Arvid Lundervold[‡] Helwig Hauser[†] Eduard Gröller[†§]

Abstract

Scientific data-sets often come with an inherent hierarchical structure such as functional substructures within organs. In this work we propose a new visualization approach for volume data which is augmented by the explicit representation of hierarchically structured data. The volumetric structures are organized in an interactive hierarchy view. Seamless zooming between data visualization, with volume rendering, and map viewing, for orientation and navigation within the hierarchy, facilitates deeper insight on multiple levels. The map shows all structures, organized in multiple hierarchy levels. Focusing on a selected node allows a visual analysis of a substructure as well as identifying its location in the hierarchy. The visual style of the node in focus, its parent and child nodes are automatically adapted during interaction to emphasize the embedding in the hierarchy. The hierarchy view is linked to a traditional tree view. The value of this new visualization approach is demonstrated on segmented MRI brain data consisting of hundreds of cortical and sub-cortical structures.

1 Introduction

Research in information visualization has many examples of visualizing hierarchical data such as trees and graphs. Scientific data often has an inherent hierarchy that is in many cases not fully exploited during visualization. In the medical domain it is often easy to describe the inherent hierarchical nature of the data. The human body can be semantically divided into several structures that have a hierarchical relationship with each other. For example the arm can be substructured into upper arm, forearm, and hand. The hand can be further divided into

*This work appeared in the proceedings of VMV 2008.
[†]Department of Informatics, University of Bergen, Norway.
[‡]Department of Biomedicine, University of Bergen, Norway
[§]Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria

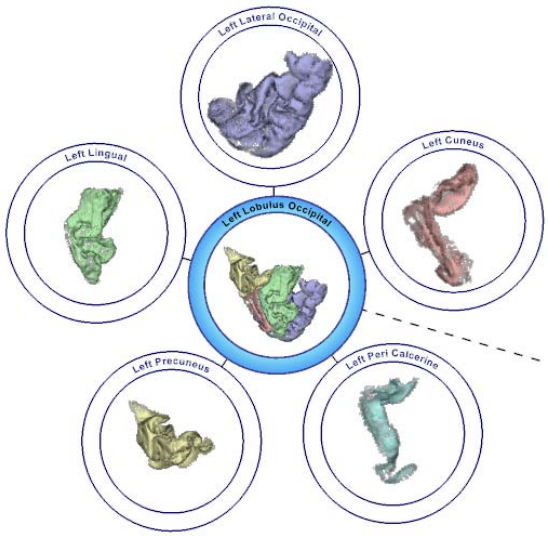
fingers and palm. Another example of a hierarchical structure, and also the one we are here focusing on, is the brain. The anatomical hierarchical subdivision of the brain starts with the separation of the left and right hemispheres, then the cortical and sub-cortical areas, followed by subdivision into different lobes, consisting of several gyri and other structures.

In medical education it is difficult to convey this 3D spatial relationship by the use of textbooks. Thus, medical students have to perform training on cadavers in order to acquire this kind of knowledge. The amount of information that is possible to extract from a textbook is to a significant amount related to the contained illustrations. The amount of knowledge gained from cutting into a real brain is also limited. Cutting open one structure to study its sub-structures will make the higher level structure unusable for further studies due to its irreversible modification. It is also possible to study brain data by looking at MRI slices, but analyzing such slices requires reasonable expertise. 3D volume visualization can help in visualizing the structures. However it is difficult to infer hierarchical and semantic information from these visualizations, especially when many structures are to be investigated.

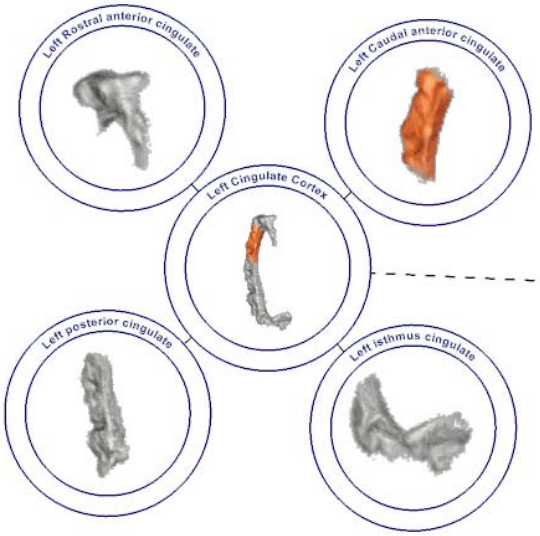
Our approach is based on two different types of data. One is 3D anatomical data from MRI, with binary segmentation masks, and the other is abstract hierarchical information inferred from the 3D data. The proposed approach in this paper tries to not only show the anatomical structure but to integrate hierarchical semantics and volume information in the same visualization. The visualization combines the field of scientific visualization with information visualization by rendering a hierarchical layout in the same view as the volume rendering. Figure 1 shows a closeup example of this combined view.

The major contribution of this work is the combined visualization of scientific volume data with inherent hierarchies. We provide a seamless interface that enables an integrated interaction between abstract hierarchies and scientific data. We do this by creating an overview map where the hierarchy of the data is represented and where it is possible to zoom in to reveal knowledge about the volume data. At the volume data level we change the visual representation of structures with auto-styling so that the hierarchical relationship between the structures becomes evident in the spatial domain. Using the novel concept of raycasting portals we are able to render more than 150 structures with volume rendering at the same time.

This paper is organized as follows: In the following section we present an overview of existing visualization techniques that relate to our work. In Section 3 we describe our approach to visualize hierarchical data and present results in Section 4. In Section 5 we discuss the results and mention future work. Finally we conclude in Section 6.



(a)



(b)

Figure 1: (a) The occipital lobe is colored to indicate the hierarchical relationship using *auto-styling*. (b) Interactive change of the visual representation of one structure in the cingulate cortex.

2 Related Work

We aim at volume rendering with the look-and-feel of medical textbooks such as the anatomical atlas by Sobotta [16]. Volume rendering has become a large field of research. The GPU-based rendering approaches that we build on are described by Engel et al. [3] and Krüger and Westermann [8]. The illustrative results are

Figure 2: Hierarchically boosted volume visualization. (a) The data basis. (b) Components of the hierarchical visualization. (c) Interaction metaphors.

produced with style transfer functions as proposed by Bruckner and Gröller [2].

Hierarchical data are easier to navigate and to gain knowledge from if an appropriate interaction metaphor and visualization is used. The evaluation done by Wang et al. [19] confirms this. Hierarchical information is often visualized as a tree. The information visualization community has done extensive research in the field of visualizing and navigating hierarchical data. One type of approach maximizes the utilization of the available screen-space, called space-filling techniques, such as tree-maps [14, 15], information slices [1], and the InterRing [21]. They also indicate size measures associated with the data. In visualizing a file-system, for example, the tree-map technique uses the size of a file or directory as a measure of the size of a structure. Since the data is hierarchical, the size coding is applied recursively and the space occupied by a parent node is subdivided by its children. The visualization used to show the parent-child relationship is depicted with rectangles inside rectangles. Similarly, information slices and InterRing use cascading circles and visualize the size measures as sector pieces.

Other techniques visualize trees without giving an indication of the relative sizes of the hierarchies. Cone Trees [13] and hyperbolic trees [10], for example, create a navigatable space with nodes in 3D. Other techniques such as RINGS [18] and Balloon trees [9] position nodes radially in 2D. The latter approaches have some similarities with our technique to lay out hierarchical data.

Interaction with and navigation of hierarchical data is also a topic of research. An example is automatic panning and zooming [20] which efficiently moves from one node to another while preserving the overview by immediately zooming out to show the context. Other approaches let the user focus on some region of interest. InterRing lets the user expand a hierarchical level of interest. The other levels are reduced automatically to accommodate the region of interest. Another approach to visualizing the region of interest especially, was suggested by Stasko and Zhang [17]. The outer or inner part of a radial visualization is used as a special area to render the region of interest. Another interesting way of performing focus+context visualization is based on non-linear magnification lenses [7].

Other visualization techniques try to make the tree and graph visualizations more sparse by reducing the number of connection lines between nodes. Examples are edge bundling [6], or changing the thickness of connection lines such as in arctrees [11]. Herman et al. [5] provide an exhaustive survey on trees and hierarchy interaction.

FreeSurfer [4] is a set of tools for the study of cortical and sub-cortical anatomy. It provides automated parcellation of the cerebral cortex and labeling of sub-cortical tissue classes in MRI volumes.

Previous work that proposes techniques to visualize the hierarchical nature of the brain has been proposed by Pommert et al. [12]. Their technique considers several different types of hierarchies. The user has to actively select a

structure, then select what type of hierarchical information is interesting from a popup menu. Our technique differs significantly from their approach. In our visualization, for example, the hierarchy is the context that the user is navigating in. When focusing on a feature more hierarchical information is automatically provided.

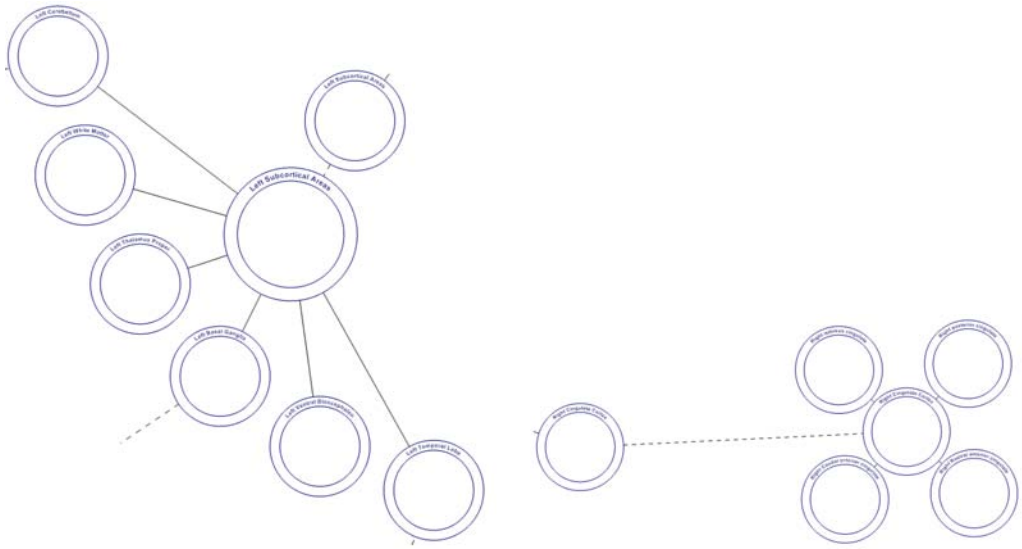
Sources that describe techniques to combine hierarchy visualization and scientific visualization in the same context are scarce. The closest solution to resemble our technique is volume rendering of segmented structures with one structure highlighted and the other structures as context.

3 Spatial Data with Hierarchical Semantics

We integrate two spaces, an abstract space with a hierarchy and a data space where the volume data is defined. We enable seamless zooming between the hierarchical model and the anatomical data in the spirit of the focus+context metaphor. We propose a tree layout of the hierarchical data where each node shows a volume rendering of the semantically associated structure and a descriptive label. We call this the *context view*. It is crucial that rendering and navigation of this view is interactive. The navigation includes zooming from the context view to the volume data and the hierarchically guided exploration of this view. Figure 2 illustrates the different aspects of our approach. Figure 2(a) shows the available data basis, in Figure 2(c) the interaction possibilities are shown, and in Figure 2(b) the visualization techniques that create the final results are shown. The visualization changes according to the user interaction. Some of the visualization techniques are only active during specific interactions, others are active during the entire interaction process.

3.1 Hierarchy in the Data

In Figure 2(a) three types of data are listed as input for our approach. The data which we visualized here is the `Bert` data-set as provided by FreeSurfer. The data is T1 weighted MRI and FreeSurfer automatically generates binary segmentation masks for many structures in the brain. This process is based on an brain atlas technique and it takes approximately 20 hours. The segmentation masks that we use are the ones generated for the cortex and the sub-cortical areas of the brain (APARC+ASEG). The segmentation masks represent small regions and structures that by themselves do not form a hierarchy. We have created a hierarchical tree that associates segmentation masks with labels and labels with groups that are semantically meaningful. For example cortical ridges, denoted as gyri, are grouped together to form larger structures called lobes. These groups are part of other groups, such as lobes that are part of the cerebral cortex. The



(a) (b)

Figure 3: Layout patterns for the context view. (a) A fan pattern, filling the area of a sector. (b) A cluster layout around a replicated group node in the center with a dashed line connecting it with its original.

hierarchical groupings stop with the hemispheres of the brain. The resulting dataset contains a hierarchical overview that anatomically and hierarchically describes the brain and its structures.

3.2 Hierarchy Visualization

The hierarchical information of the data is visualized in two ways. First, a context view is generated that illustrates the hierarchical structure. A node-link diagram is used. Every node has links to its children and to its parent and all nodes are labeled. The automatic layout scheme attempts to create a context view that takes advantage of symmetries to support the orientation of the user. Second, child and parent nodes are displayed close to each other. A unique coloring of sub-structures indicates the hierarchical structure. An example of this can be seen in Figure 1(a).

We have defined semantics for the hierarchical information associated with the data. A leaf node is a segment node. These nodes have a direct correspondence with a segmentation mask. Nodes with children are group nodes, and group nodes that only contain leaf nodes are called leaf-clusters. This semantic is used for context view creation, auto-styling, and volume rendering. The leaf nodes of a leaf-cluster are shown in Figure 3(b). In the same figure the two nodes connected with a dashed line are group nodes.

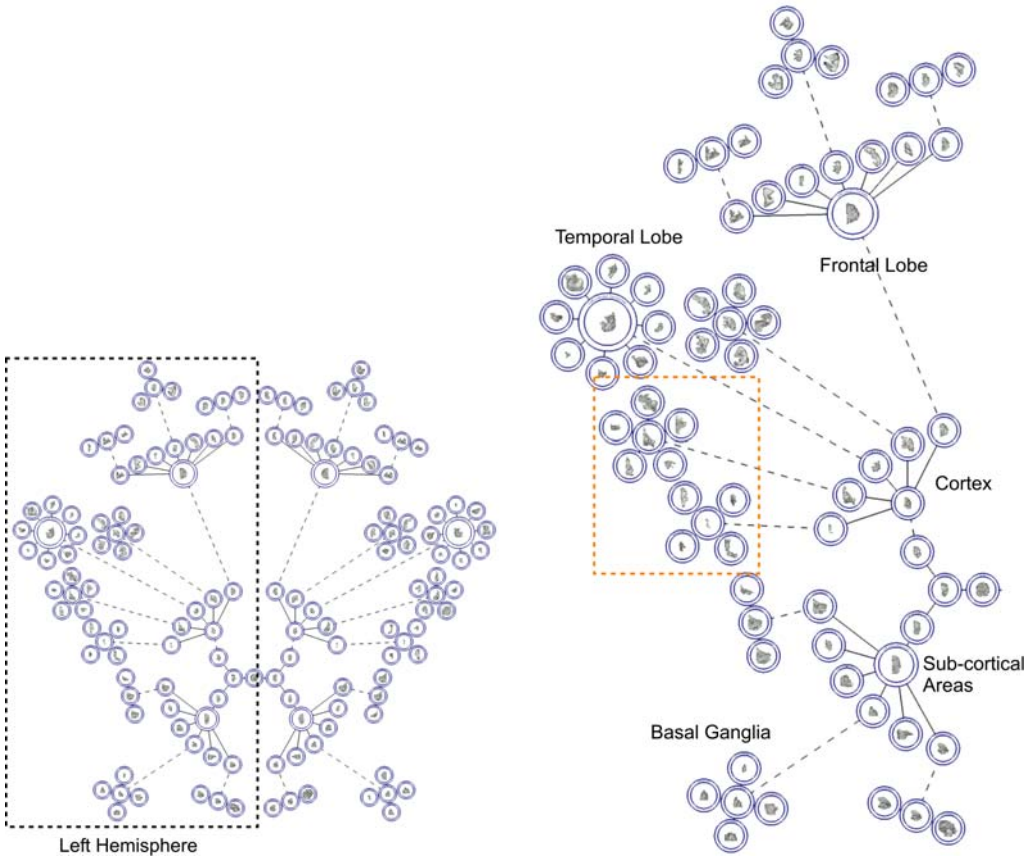
To optimize screen-space utilization we place structures as close to each other as possible, when we create the context view. In addition, structures should be positioned in such a way that the hierarchical relations are self evident. The user should fast and easily recognize the different structural features as generated in the context view.

The design choices of the context view are derived from the hierarchical nature of the data (Figure 4(a)). We place structures in a radial pattern and assign sub-structures to fractions of the sectors that are occupied by higher level structures. The left half and the right half of the layout correspond to the left and right hemispheres of the brain. Each quadrant represents a high level feature. These features are the cortex in the upper quadrants and the sub-cortical areas in the lower quadrants. Sub-structures of these features are given as fractions of these four sectors. Groups get a fraction of their parent's sector based on the number of children.

Some of the structures at the leaf level in the hierarchy have many siblings. In the case of leaf-clusters we want to minimize the occupied screen-space. Nodes in a leaf-cluster are positioned around a central point without any overlap. For helping the user to keep the context in mind, the group node of the leaf-cluster is replicated in the center of the cluster (Figure 3(b)). The clustering reduces the area occupied by sibling nodes relative to the sector size. If we do not cluster the sibling nodes in this manner the space needed to draw all the sibling nodes increases, causing the nodes to move further away from the original parent node. Groups that only have group nodes or a combination of group nodes and leaf nodes as children are placed in a fan pattern, positioned at a distance where nodes do not overlap. This means that the nodes must be moved to a distance where all nodes can be positioned on an arc within the sector bounds. The group node is in this case replicated as well but it is positioned between the fan and the original group node (Figure 3(a)). At the two highest levels. i.e., the brain and the two hemispheres, we do not replicate the group nodes. The size of the rendered replicated group node is adjusted depending on the number of siblings. In a leaf-cluster, for example, the replicated group node increases in size when the number of siblings creates a circle that is much larger than the minimum node size (Figure 8).

The position where we place a replicated group node is also the position we use to bundle the connection lines between parent and children. In Figure 3 this can be seen as the connection lines from leaf nodes and group-nodes to the replicated group node of their parent. This makes the overview less cluttered. Drawing one line between a group node and a replicated group node and then one line from each child to the replicated group node is more space efficient than one line per child node to the original group node.

The Figure 4(a) shows the complete hierarchy of the brain with every node rendered in a ring. The ring is rendered as two concentric circles with different



(a) (b)

Figure 4: A context view (a) showing the entire hierarchical layout with a close up below (b). The right hemisphere is indicated in (a) as the dashed rectangle. Some features have been labeled in the figure to indicate their position. The close up of the region enclosed by the orange dashed rectangle is shown in Figure 1.

radii. The orange dashed square in Figure 4(b) represents the zoomed area given in Figure 1. In this closeup we can see most of the different types of visualizations from our approach. In Figure 1 there are two structures organized in two leaf-clusters, i.e., the occipital lobe and the cingulate cortex. When a node is highlighted, its ring is rendered in blue gradients. In case of replicated group nodes the highlighting is done for both nodes. Each node has an associated label which is placed on a curve on the top half of the ring.

When the user is looking at the context view, lines provide hierarchical information, such as child-parent relationships. The algorithm draws lines recursively between parent nodes and child nodes. The line is drawn from the border of the ring. In the case of replicated nodes, dashed lines are drawn between the original node and the replicated node. This is shown in Figure 1 and Figure 4.

An intuitive way of interaction with the context view is the possibility to focus on nodes. There are two types of direct interaction which result in focusing, i.e., hovering over a node and selection. When the mouse hovers over a node the node will initiate *auto-styling*, described in Section 3.4. Selection is done through clicking on a node. This centers the view on the node and enables manual selection of the style of that structure (Section 3.5).

3.3 Raycasting Portals

In the center of every node a volume rendering of the associated structure from the hierarchy is shown. This depiction of the volume data is generated by a GPU-based volume raycasting technique [8]. A proxy geometry is used to render the cubical shape of the volume data. We use the 3D texture coordinates to render colors that we use as a map into the volume data. The structures to visualize usually occupy only a sub part of the volume. If we only render the voxels included in the segmentation mask, we can significantly improve rendering time. We calculate the bounding boxes for all segmentation masks and the bounding boxes for all nodes higher in the hierarchy and use this to reshape the proxy geometry. We offset the texture coordinates so that they map to the coordinates of the bounding box of the structure. The aspect ratios of the proxy geometry are adjusted to match the aspect ratios of the bounding box. An example of the reshaped proxy geometry for the result seen in Figure 1(b) is given in Figure 5 where the raycasting starting position values are depicted.

To change the visual representation of the raycasted structures, we use style transfer functions [2]. Style transfer functions utilize lighting information as acquired from orthogonally projected lit spheres. This technique makes it simple to achieve view-aligned lighting. With a style transfer function it is easy to switch from simple diffuse Lambert shading to Phong like shading by using different lit spheres. This technique enables us to simulate the non-photo realistic illustration style of medical anatomy illustrations like the ones by Sobotta [16]. See Figure 7 for examples.

We also need a mechanism to control what styles the raycaster should use for a specific substructure. Our solution to this is to use raycasting *portals*. Usually for GPU accelerated volume raycasting, a full-screen quad is rendered to the screen which is also a trigger for the GPU program to generate pixels. Instead of creating a full-screen quad we render a quad for every structure that we need raycasting for. A quad is centered on every node and the size of the quad is equal to the bounding square of the node drawn at that node. We call these quads raycasting portals. The main feature of these portals is that we can now communicate portal-specific rendering parameters to the GPU program via a shader uniform variable. The uniform variable contains a list of styles for the structures that should be visible and highlighted, and a zero reference to all

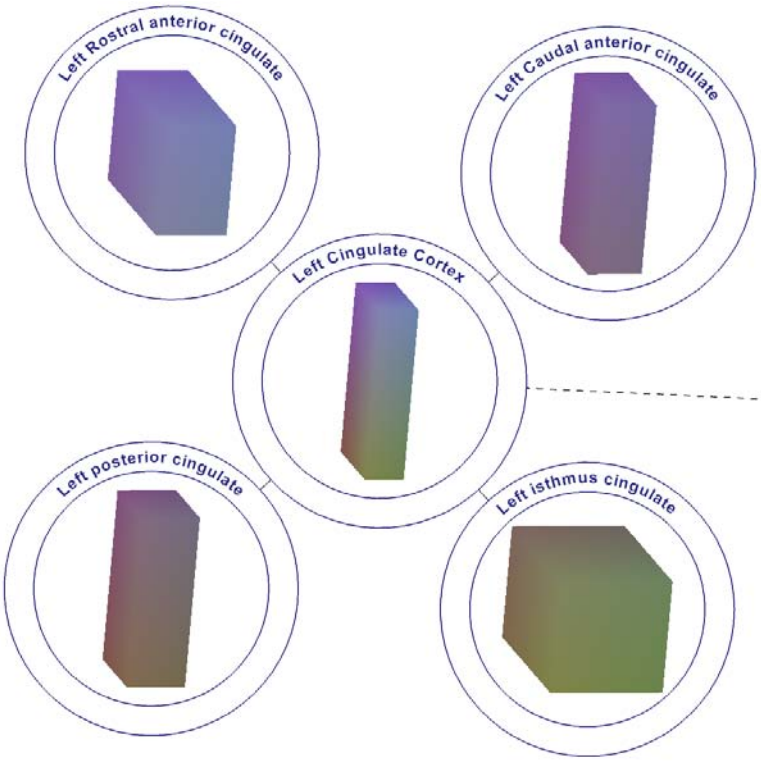


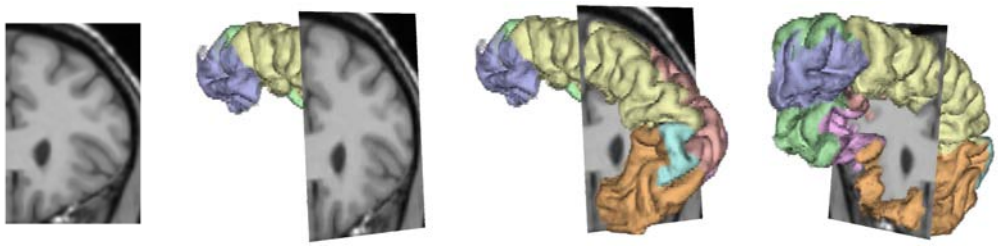
Figure 5: A visualization of rendered proxy geometries with the starting positions for the raycasting.

structures that should be invisible.

To further increase the amount of information that is associated with the hierarchical elements and their volumetric nature, we have additionally implemented axis-aligned slicing. This visualization gives the user the opportunity to study the underlying MRI data and not just the raycasting of the structures. Figure 6 illustrates the combination of 3D and 2D information. Some of the possible combinations of slicing and volume rendering are illustrated in Figure 6.

3.4 Hierarchical Styling

The hierarchical organization of the data is used when we convey the hierarchical arrangement between sub-volumes. We change the visual representation of structures to clearly illustrate the hierarchical relationships. We have implemented two ways of interacting with the visual representation of structures to achieve two different goals. The first goal is to visualize the hierarchical relationship between the parent and the child. This is done by showing how the parent node is composed of the child nodes by uniquely coloring each structure. The second goal is to show how a single structure or multiple structures are spatially located in all



(a) (b) (c) (d)

Figure 6: Axis-aligned slicing of the frontal lobe: (a) direct view of a slice in the z direction. (b) structure and slice, features in front of the slice have been removed. (c, d) structure and slice from two different view-points (no removal of structures).

applicable hierarchy levels. This is achieved by the user interactively setting the style for a structure. The first approach is shown in Figure 1(a) and the second one is shown in Figure 1(b).

Our goal is to provide the user with multiple ways of seeing how the hierarchical structures are organized. One way of doing this is to color each structure uniquely so that they are easy to differentiate from each other. When the mouse hovers over a node, a feature called auto-styling is initiated. Auto-styling is the visual result of applying pre-defined style transfer functions to structures. This feature simplifies the navigation through the hierarchy. In addition it enhances the mental image the user has of the 3D structures of the brain. We have defined eight perceptually different styles based on pastel versions of red, green, blue, cyan, magenta, yellow, orange, and a darker blue. These colors have been selected based on contrast and lightness. The pre-defined styles are customizable.

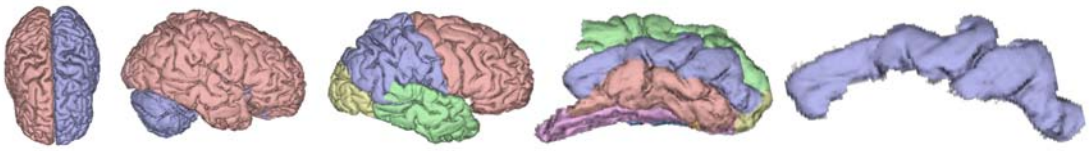
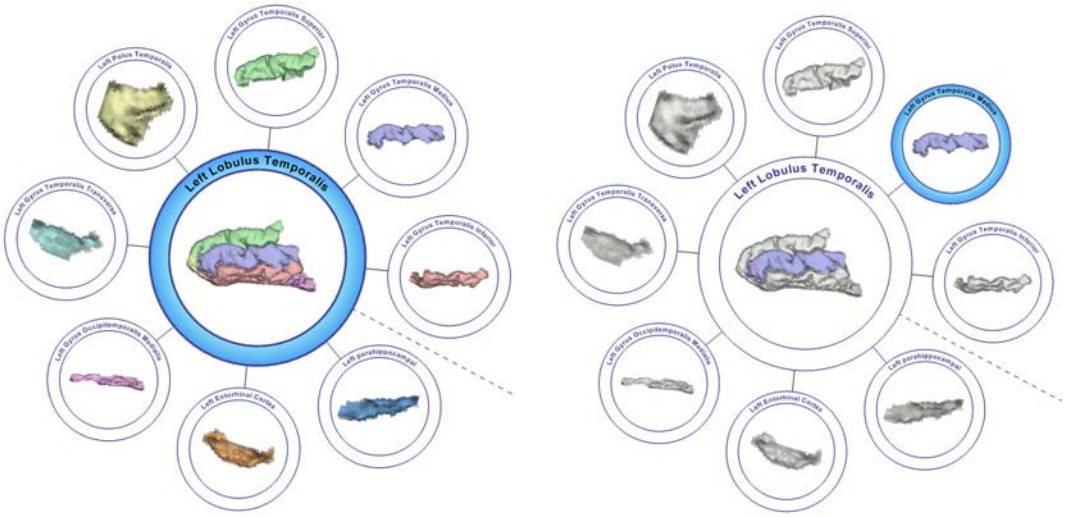


Figure 7: Auto-styling of hierarchically linked structures. From left to right: entire brain, right hemisphere, right cortex, right temporal lobe and right medial temporal gyrus.

If the mouse hovers over a group node, all its children are set to one of the predefined styles. The children are assigned styles in the same order as they are defined in the hierarchy. The style applied to a child is also used in the visual representation of the group node. This can be seen in Figure 1(a) and in Figure 8(a). In Figure 7 it is possible to see where the right medial temporal gyrus is located in the right temporal lobe using this technique.



(a) (b)

Figure 8: (a) Auto-styling of nodes in a leaf-cluster when the group node is selected. (b) Auto-styling of nodes in a leaf-cluster when the leaf node is selected.

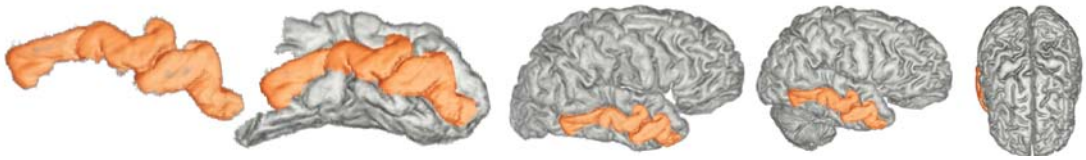


Figure 9: User-specified styling of a selected structure. From left to right: right medial temporal gyrus, right temporal lobe, right cortex, right hemisphere and entire brain.

If the mouse hovers over a child node, this node is displayed using the same pre-defined style as in the previous case, but the group node is displayed differently. The group node is displayed using the group's default style (defaults to grey). The selected structure is the only structure that is displayed with a different visual representation. This can be observed in Figure 8(b) where the selected structure can be seen in the group in light blue color.

Tracking a specific sub-structure in several different hierarchical levels is also possible. The user can change the style of a structure and then move up in the hierarchy to observe where the structure is located relative to higher levels in the hierarchy. Figure 9 illustrates this visual enhancement. The user has changed the style of the medial gyrus part of the temporal lobe. Moving up the hierarchy, the structure is now highlighted with the user selected orange style.

3.5 Interaction and Navigation

We enable the user to navigate by panning the data, hierarchically guided navigation, seamless zooming from the overview to the volume rendering, and by rotation of volume geometry. The user can hover over nodes to initiate auto-styling or manually set the style of interesting structures so they can easily be tracked. When exploring the data, the user can move the entire view or zoom in and out. Exploring the hierarchy in a zoomed-in manner can be a tedious task. A lot of dragging with the mouse is necessary to move around. Therefore we have implemented guided navigation that helps the user to move between nodes. If the user selects a node, the view centers on that node. This also means that zooming in and out is now centered on this node. If the node is a leaf node then the guided navigation will move the focus to the parent node. If the user initiates guided navigation on a replicated group node then the focus changes between the group node and its replica.

When the user is at an abstraction level, which shows volume rendered structures, rotation of the structures is possible. Rotation applies to all structures so that there is always a coherency between the views of the volumetric data. The user interface also lets the user change the visual representation of a selected structure so that it is possible to see where a feature is located in higher level structures.

4 Results

Seamless zooming from the contextual overview down to the data at the lowest hierarchical level can be observed in Figure 1 and Figure 4. In Figure 4(a) the complete overview is shown. The left part of this overview represents the left hemisphere and the right part represents the right hemisphere. Figure 4(b) is the left hemisphere only. At this level it is possible to see thumbnail sized volume raycastings of the structures. As the labels on the right hemisphere in the figure indicate, the upper part is the cortex and the lower part represents the sub-cortical areas. Zooming into the dashed orange rectangle we get the information as shown in Figure 1. In this figure we start to see details of the volume renderings and labels are readable. It is for example possible to see where the lingual is located in the occipital lobe (green structure in Figure 1(a)).

During inspection of the volume data at the lowest level of data exploration, it is possible to enhance the volume rendering with slicing so that the original data might be inspected. Figure 6 shows this concept. The user can choose to only show the slices. It is also possible to mix the two types of visualization and in this way get a higher level of understanding of the 3D nature of structures relative to 2D slices.

Auto-styling is highlighted in Figure 7. This Figure shows how the styling

looks like at the different hierarchy levels if a user would start at the brain and navigate down to the medial gyrus of the temporal lobe. Going the other way, from the gyrus, and seeing how this structure is positioned relative to the higher level structures is illustrated in Figure 9.

We have implemented our approach in Java using OpenGL on a GeForce 8800 GTS. We used the OpenGL Shading Language extensions, texture arrays and integer texel look-ups, provided in the ShaderModel 4.0 specification.

The `Bert` data-set that we have visualized to create the results in this work is T1-weighted 3D MRI data at an isotropic resolution of 256^3 with 1 μL (microliter) voxels. This data-set has been run through the automated segmentation workflow of FreeSurfer. The surface reconstruction and volume labeling has produced approximately 80 sub-cortical and 68 cortical segmentations. We have in addition created a data-set that describes the hierarchical semantics of the segmentations. Based on these three data-sets we have generated a contextual overview that illustrates the hierarchy of the data. We render the MRI data using volume raycasting.

The context view without any volume raycasting renders at around 50 frames per second. Adding volume raycasting reduces the rendering speed to around 10 frames per second. The performance depends on how many pixels a structure covers, the number of voxels in the rendered structure and the amount of transparency defined in the style transfer function. To keep the interaction fast the raycasting is speeded up during interaction by doubling the sampling distance. This effectively doubles the rendering speed.

5 Discussion and Future Work

The presented approach to hierarchical brain visualization has an immediate use in education. Learning neuroanatomy and neurophysiology has always been regarded as challenging for medical students. This is due to the inherently complex and abstract structure of the nervous system, and to the intricate three-dimensional organization of the human brain. Our approach allows to visualize how the brain's components fit together, both in a strictly anatomical setting, and also in a functional-hierarchical manner. A daunting task is to understand which parts of the brain are in connection with which others and how they function together. The here presented approach has the potential to increase the efficiency of learning and ease the process of comprehension. Furthermore in radiology, students are required to relate planar images in different orientations to recorded volumes from modalities such as Computed Tomography and MRI. Our approach provides a way to investigate how the cut-planes actually represent parts of the volume. This will help students to understand the process of planar imaging, which is something many find difficult, and help themselves relate the

two-dimensional image to the actual volume in real-time.

In a clinical setting it is important to convey subject-specific visualizations from the data. In dementia research, for example, it is of interest to compare volumetrics between a statistically normal brain and probable dementia patients. This comparison can be implemented in the hierarchical visualization to draw attention to structures that deviate, or not, from the normal population. We also foresee that including data obtained from other imaging sources such as functional MRI and diffusion tensor imaging can have a potential use in a clinical setting. Further research and evalutation of these approaches is left to future work.

6 Conclusion

We have presented an approach to visualize hierarchical volume data and enable hierarchy-based interaction. We have described how to generate the context view that enables a seamless navigation from the abstraction to the data. The visualization of structures in focus is automatically changed to reflect the hierarchical nature of the data.

Based on feedback from the medical side, our new visualization concept is considered promising and useful for medical education, especially for teaching radiologists. They usually look at 2D slices only and it is very hard for students to grasp the 3D structure of the structures observed on planar slices. With our approach they can select a portion of the data according to the hierarchical structure, and avoid that they are overloaded with the entire data-set. The integrated slice rendering gives the student a correspondence between 2D and 3D.

In general we do not see any difficulties to adapt our approach to other hierarchical structures such as the bones of the human hand. We think that there are other scientific domains with data containing hierarchies that would benefit from such an approach, also. An outlook for the future in hierarchical brain visualization comes from our medical collaborators. They observe that the anatomical structure of the brain is rather artifical and in many cases does not fit the functional hierarchy. Visualizing functional hierarchies will move the applicability of this tool from educational use to clinical use.

References

[1] K. Andrews and H. Heidegger. Information slices: Visualising and exploring large hierarchies using cascading, semi-circular discs. In *LBHT, IEEE InfoVis '98*, pages 9–12, 1998.

[2] S. Bruckner and M. E. Gröller. Style transfer functions for illustrative volume rendering. *CGF*, 26(3):715–724, Sep 2007.

[3] K. Engel, M. Hadwiger, J. M. Kniss, C. Rezk-Salama, and D. Weiskopf. *Real-time Volume Graphics*. A. K. Peters, 2006.

[4] Freesurfer. http://surfer.nmr.mgh.harvard.edu.

[5] I. Herman, G. Melancon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE TVCG*, 6(1):24–43, 2000.

[6] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE TVCG*, 12(5):741–748, Oct 2006.

[7] T. Keahey. The generalized detail-in-context problem. *IEEE InfoVis '98*, pages 44–51, 1998.

[8] J. Krüger and R. Westermann. Acceleration techniques for GPU-based volume rendering. In *IEEE Visualization 2003*, pages 287–292, 2003.

[9] G. Melancon and I. Herman. Circular drawings of rooted trees. Technical Report INS-R9817, CWI, Amsterdam, Netherlands, 1998.

[10] T. Munzner. H3: laying out large directed graphs in 3D hyperbolic space. *IEEE InfoVis '97*, pages 2–10, 1997.

[11] P. Neumann, S. Schlechtweg, and M. Carpendale. Arctrees: Visualizing relations in hierarchical data. In *EuroVis '05*, pages 53–60, 2005.

[12] A. Pommert, Schubert, Riemer, Schiemann, Tiede, and Höhne. Symbolic modeling of human anatomy for visualization and simulation. In *Vis. in Biomed. Comp.*, volume 2359, pages 412–423. SPIE, 1994.

[13] G. Robertson, J. Mackinlay, and S. Card. Cone trees: animated 3D visualizations of hierarchical information. In *Proc. CHI '91*, pages 189–194. ACM, 1991.

[14] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1):92–99, 1992.

[15] B. Shneiderman and M. Wattenberg. Ordered treemap layouts. *InfoVis '01*, pages 73–78, 2001.

[16] J. Sobotta. *Sobotta Atlas of Human Anatomy*. Lippincott Williams & Wilkin, 2001.

[17] J. Stasko and E. Zhang. Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. *IEEE InfoVis 2000*, pages 57–65, 2000.

[18] S. T. Teoh and K.-L. Ma. RINGS: A technique for visualizing large hierarchies. In *GD '02*, pages 268–275, 2002.

[19] Y. Wang, S. Teoh, and K.-L. Ma. Evaluating the effectiveness of tree visualization systems for knowledge discovery. In *EuroVis '06*, pages 67–74, 2006.

[20] J. Wijk and Nuij. Smooth and efficient zooming and panning. *InfoVis '03*, pages 15–23, 2003.

[21] J. Yang, M. Ward, and E. Rundensteiner. Interring: an interactive tool for visually navigating and manipulating hierarchical structures. *IEEE InfoVis 2002*, pages 77–84, 2002.

Paper IV

Interactive Illustrative Visualization of Hierarchical Volume Data

Jean-Paul Balabanian* Ivan Viola* Eduard Gröller*†

Abstract

In scientific visualization the underlying data often has an inherent abstract and hierarchical structure. Therefore, the same dataset can simultaneously be studied with respect to its characteristics in the three-dimensional space and in the hierarchy space. Often both characteristics are equally important to convey. For such scenarios we explore the combination of hierarchy visualization and scientific visualization, where both data spaces are effectively integrated. We have been inspired by illustrations of species evolution where hierarchical information is often present. Motivated by these traditional illustrations, we introduce integrated visualizations for hierarchically organized volumetric datasets. The hierarchy data is displayed as a graph, whose nodes are visually augmented to depict the corresponding 3D information. These augmentations include images due to volume raycasting, slicing of 3D structures, and indicators of structure visibility from occlusion testing. New interaction metaphors are presented that extend visualizations and interactions, typical for one visualization space, to control visualization parameters of the other space. Interaction on a node in the hierarchy influences visual representations of 3D structures and vice versa. We integrate both the abstract and the scientific visualizations into one view which avoids frequent refocusing typical for interaction with linked-view layouts. We demonstrate our approach on different volumetric datasets enhanced with hierarchical information.

1 Introduction

Datasets coming from scientific domains are usually defined with respect to a spatial frame of reference. Examples are volumetric data acquired using computed tomography, seismic acoustic measurements of geological structures, or weather

*Department of Informatics, University of Bergen, Norway.
†Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria

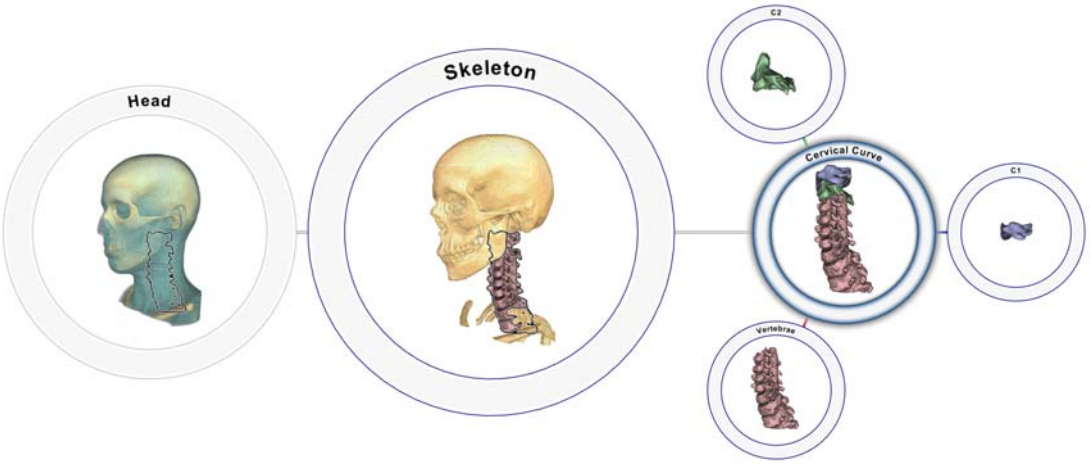


Figure 1: Hierarchical visualization of segmented head and neck. The cervical curve is focused by showing its relative position in the neck and highlighting its substructures.

simulation runs. These datasets represent a phenomenon in reality and they are analyzed with respect to their spatial structural arrangement. Increasingly for such phenomena additional data is available in an abstract space, for example, depicting relationships between various structures contained in the data. Essentially the same real-world phenomenon can be studied in two entirely different spaces.

A good example is the structure of the human body. The human anatomy can be given as 3D volumetric data. On the other hand the body consists of various hierarchically organized sub-systems like nervous, muscular and vascular systems. These systems define abstract relationships between body parts. The relationships are crucial to better understand processes in the human body. In the human motor system, for example, it is very important to analyze both, the relationships and the shape of skeletal structures.

Another example is evolution of lifeforms. In this example it is interesting to study: a) the individual representatives of a certain evolutionary stage and b) the hierarchy of the evolution as such. A similar example is genealogy. These examples are often represented in illustrations as hierarchies where nodes are augmented with particular representatives, such as the illustrations in Figures 2 and 3. In fact such a visual representation is not limited in use for evolution and genealogy. It can be used for any type of data that contains both spatial and hierarchical structuring, such as the human body from the first example. Even though relations in the human body are entirely different than relations in evolution or genealogy, the type of illustration where a hierarchy graph is augmented with node representatives enables studying particular data in both spaces simultaneously. An early example of using such an illustration concept

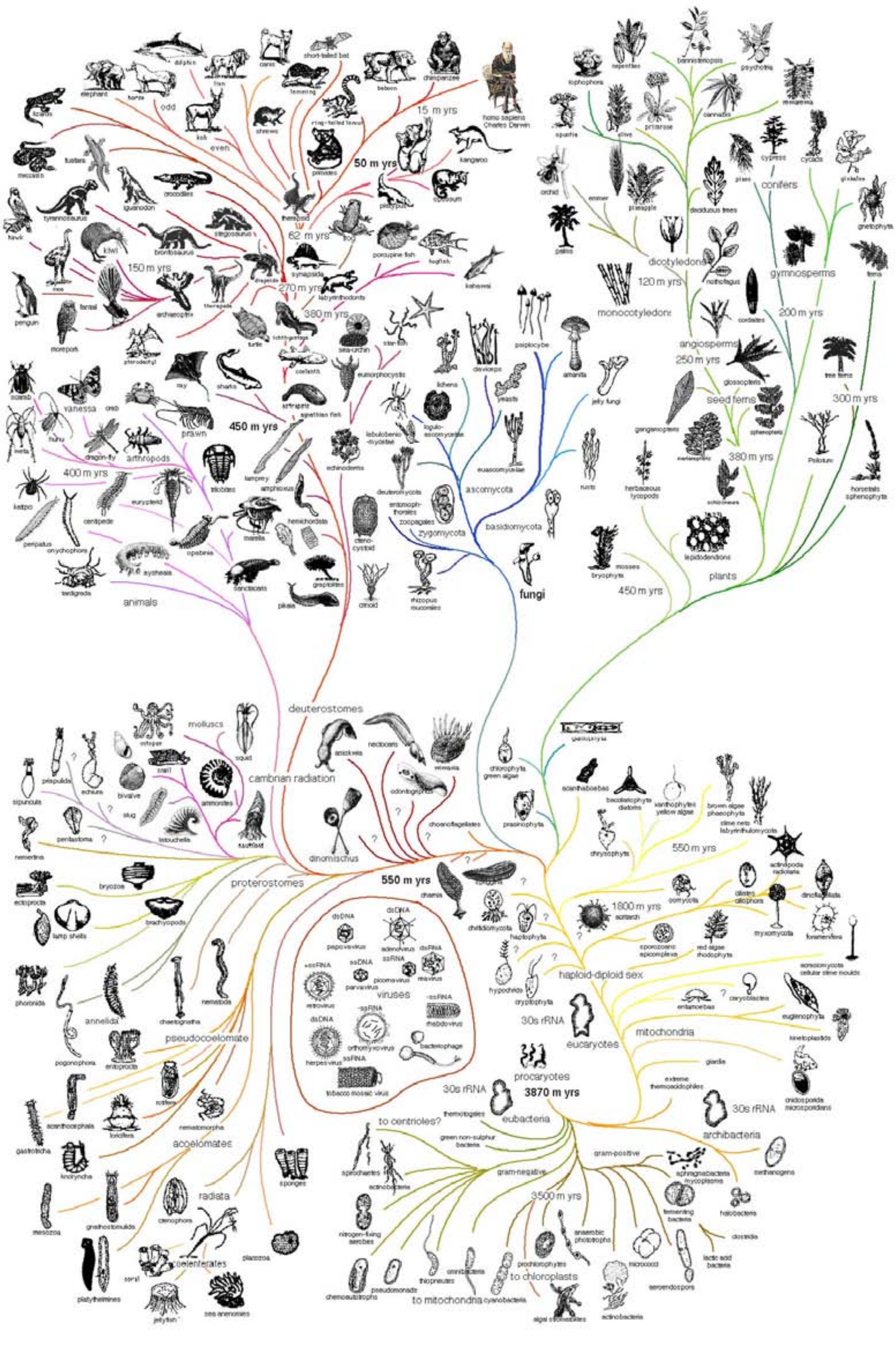


Figure 2: A complex evolution tree for organic lifeforms.

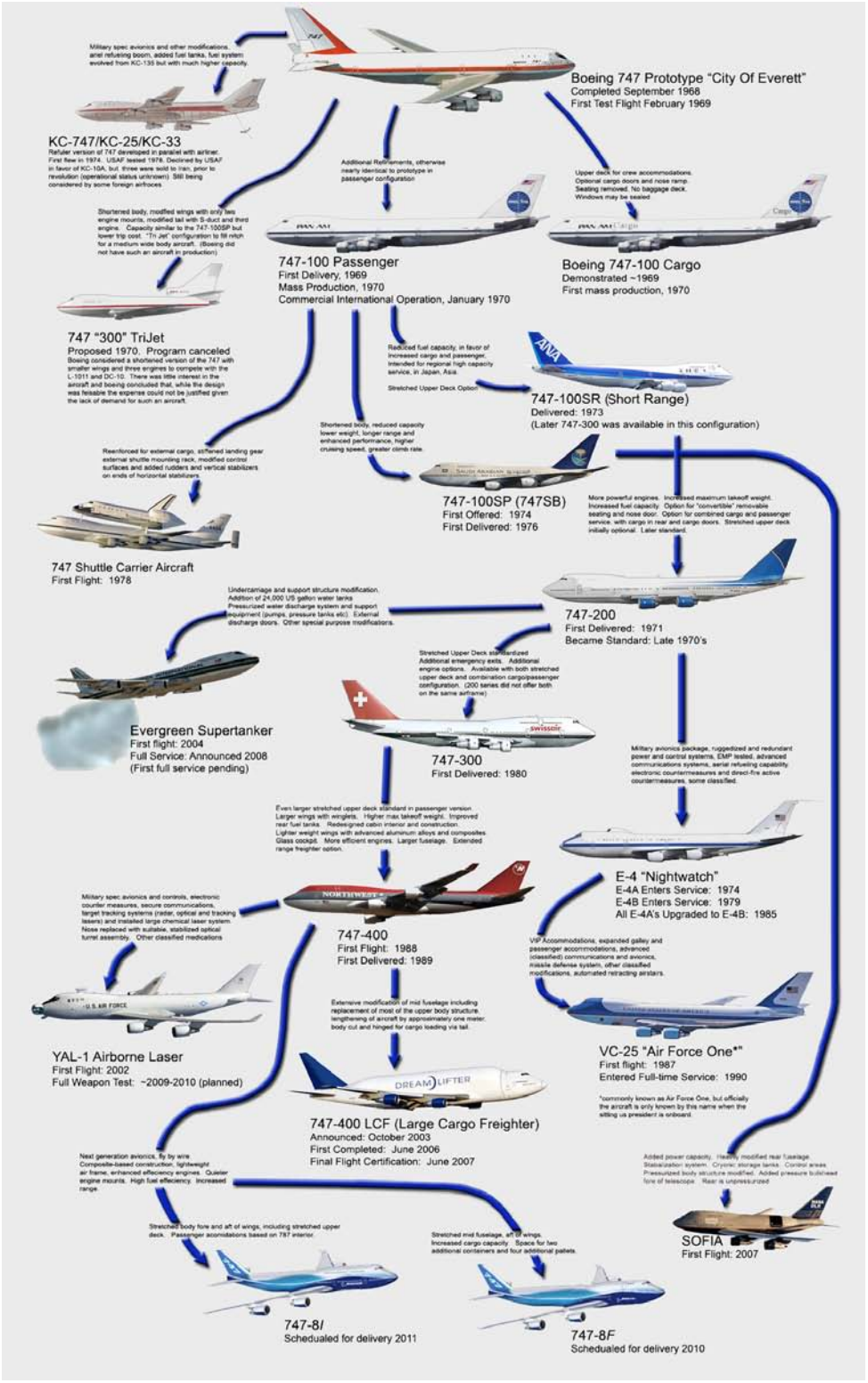


Figure 3: A 'genealogy' tree for the Boeing 747 series of airplanes.

can be seen in Figure 1.

Current visualization technology enables the user to study the spatial arrangement of scanned human anatomy using techniques from volume visualization. Structures can be visually represented using slicing or volume rendering. To analyze these structures, visualization technology offers various interactions such as defining which and how data values are shown (e.g., by using transfer functions), or from which viewing angle they are shown (e.g., by defining the viewpoint position). With such visualization approaches, the data which is defined in both spaces, in the spatial and the abstract domain, will be *projected* to the spatial domain and only the spatial characteristics will be visually conveyed. In this paper we use the words space and domain interchangeably when we refer to the spatial and abstract origins of data, interaction and visualization.

Abstract data visualization is another way to represent this type of data. Structures can be depicted by techniques developed over the years in information visualization, for example, through graphs given as node-link diagrams. For each specific category of graphs various layouts have been proposed with well defined interactions thereon. Such a representation clearly communicates information about processes and relationships. However the spatial aspect of the data is missing due to the *projection* into the abstract space only.

In visualization to convey both aspects, i.e., the spatial arrangement of structures and the abstract relational information, one possibility is to employ linked views. In such a visualization setting, both spaces are shown in separate views, and both spaces are analyzed with separate interactions. The views are linked in the sense that manipulating one view will affect the other view as well. Linking and brushing is an example where the interactively selected subset in one view will also be highlighted in the other view. The separate views, however, require switching between domains and require refocusing of the user from one space to another even if linking is present.

We believe that a stronger integration of spatial and abstract domains can lead to a better overall understanding of the studied real-world phenomenon. This is supported by illustrations depicting evolution and genealogy. Our approach is following this idea from illustrations, we display a graph as a guiding structure for understanding relationships and integrating the spatial characteristics of the data within the graph. The main contribution of this paper stems from this static illustration concept and develops an interactive integrated visualization approach. We define a set of interactions and visualizations that tightly integrate the distinct domains the data is defined in.

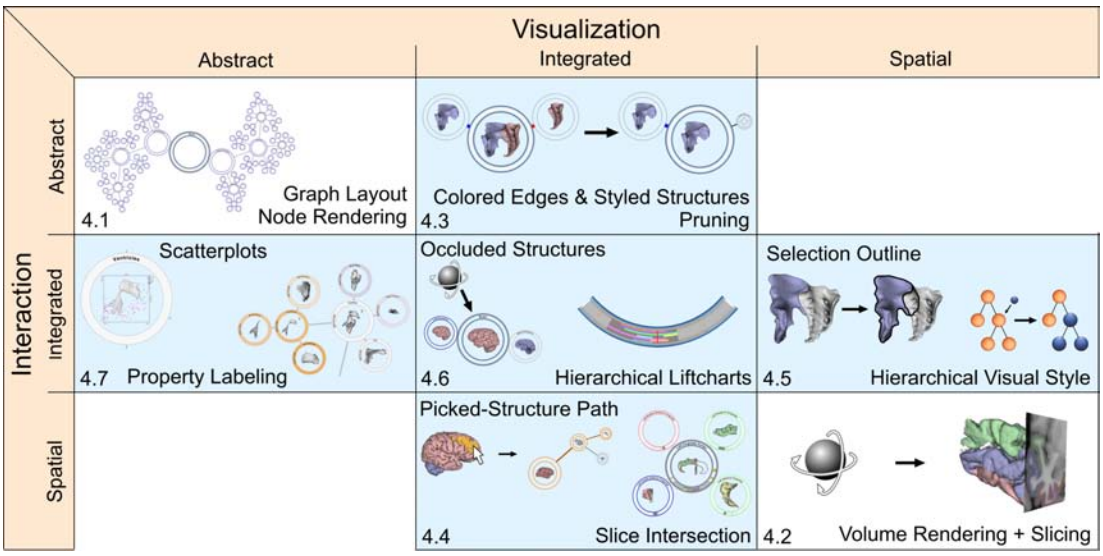


Figure 4: Matrix depicting combinations of interactions and visualizations defined for the abstract, the integrated, and the spatial domain.

2 Related Work

For visualizing the spatial characteristics in our integrated approach we rely on existing technology developed in the last decade in volume visualization. The GPU-based rendering approaches that we build on are described by Engel et al. [4] and Krüger and Westermann [9]. The illustrative results in our work are produced utilizing style transfer functions as proposed by Bruckner and Gröller [2]. In addition to volume rendering, we visualize the spatial data by slicing. We augment the slicing with LiftCharts, proposed by Tietjen et al. [15]. Alongside the slicing, a chart is visualized that shows the extent of segmented structures in the slicing direction. This gives a good indication of structure location and relation to the slicing plane and other structures. We modified this idea to realize our slice bars in Section 4.

Hierarchical data are easier to navigate and to gain knowledge from if an appropriate interaction metaphor and visualization is used. The evaluation done by Wang et al. [16] supports this statement. Hierarchical information is often visualized as a tree. The information visualization community has done extensive research in the field of visualizing and navigating hierarchical data. Herman et al. [8] provide a broad survey on trees and hierarchy interaction.

The problem of integrating data from different spaces is one of the topics that focus+context research [7] has addressed in visualization. Such integration is mostly addressing visualization of data originating from essentially the same domain. An example could be data at different scales or from different acquisition modalities. Our approach, as compared to focus+context techniques, aims at

integration of strongly different domains.

For volumetric datasets the relationship between structures is increasingly being studied using visualization. Recently, a relation-aware volume exploration [3] approach has been proposed. It defines region-connection calculus and builds for each tagged volumetric dataset a set of relations into a relation graph. The calculus and graph are used for steering visualization parameters such as viewpoint settings or visual representations. The paper is focusing on data similar to ours, but the approach is realized through linked views unlike our integrated visualization approach. Integrating abstract information into 3D spatial rendering has been proposed by Pommert et al. [14]. They integrated popup menus into the 3D rendering. These menus provide different possibilities to change the visual representation of hierarchical structures and convey hierarchical information about the selected structure. Another approach to visualize 3D structures using abstract data was proposed by Li et al. [10]. They describe an exploded view visualization that relies on hierarchical information derived from the 3D spatial structuring. The hierarchical information is used to steer the explosions. Integration of abstract visualization and spatial visualization using graph rendering and volume rendering has been proposed before [1]. They created a simple integrated visualization of a fixed graph layout with volume rendering inside the nodes. While there are similarities between these works, the previous work focuses on applicability in brain-imaging training, while we provide here a conceptual foundation for multi-space integration and a categorization of the newly developed techniques.

3 Integrated Visualization and Interaction Space

To effectively convey information about datasets defined over a spatial and an abstract domain, both domains have to be present in the visualization. In our work we focus on a strongly integrated visualization of spatial and hierarchical information. Unlike traditional approaches, where in one view only one domain is represented, we propose a tightly integrated display. In the process of merging these two domains we have chosen to use the abstract-domain representation through a graph as the guiding structure. We augment each hierarchical node with spatial information and aggregated information from both domains. This integrated visualization requires new visual and interaction means to effectively realize the visual dialog between the two *merged* domains.

In this work the integration is steered by the graph drawing. The abstract data is used to create a structure to present both the abstract and spatial data. It would also be possible to envision an approach that uses the scientific-visualization space as the embedding space. In Figure 5 we have sketched the imaginary interpolation between the two spaces that are part of the visualization, i.e., the abstract and spatial space. The red circle indicates where this work is located

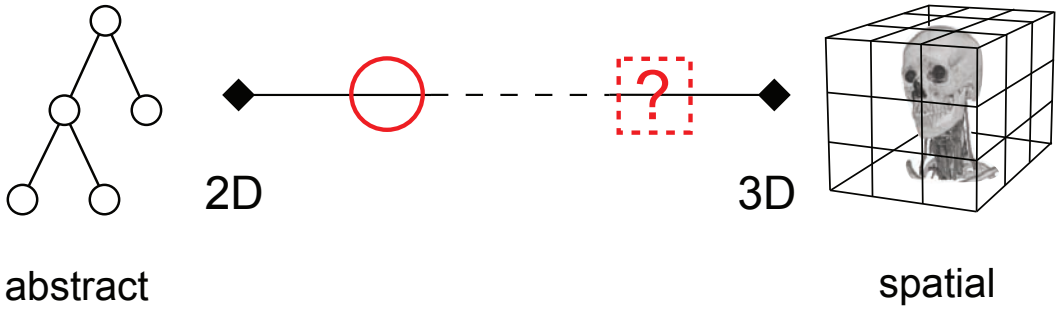


Figure 5: Imaginary *interpolation* between the abstract domain and the spatial domain. The red circle indicates where our work is contributing.

but using the scientific visualization as the embedding space will result in a visualization located in the dashed square. Such an integrated view might be an exploded view in 3D space where the abstract hierarchical relationships are indicated through arrows.

The proposed visualization inherits visualizations and interactions used previously for each respective domain separately. Essentially we can now classify three categories of visualizations and interactions: abstract, integrated, and spatial. An abstract visualization is, for example, the display of a graph using a space-filling layout. An interaction on this abstract data representation is focusing on a node which invokes a change in its size or color. Similarly, a purely spatial domain visualization is direct volume rendering. A spatial interaction will be a manipulation of the viewpoint for example.

Apart from visualizations and interactions defined exclusively for one particular domain, our integrated approach especially focuses on integrated visualizations and interactions. An integrated interaction means that a particular interaction invokes visual changes in both, now integrated, spaces simultaneously. For example, when a viewpoint is changed in the spatial domain, spatial data is rendered from the new viewpoint. We can then perform occlusion testing to check if from the new viewpoint a structure is occluded by other objects. Occluded structures are marked as such by coloring the corresponding graph nodes. This makes it easy from an overview perspective to identify occluded structures. An example that incorporates integrated interaction is hierarchically changing the visual representations of structures. This interaction must be performed in both domains. Defining the style and color is performed in the spatial domain while deciding what hierarchical node to propagate the change from requires interaction with the abstract domain.

We give an overview on possible combinations of spatial, integrated and abstract visualizations and interactions in the matrix in Figure 4. There are essentially nine possible combinations. The traditional single-domain visualization

and interaction approaches are shown in the top-left and bottom-right cells.

More interesting are the new integrated visualizations and interactions depicted in the blue cells. An interaction that is defined in only one domain can invoke an integrated visualization. An example is node pruning where the graph layout is affected and the spatial visualization shows only those 3D features whose nodes have not been pruned. There are integrated and spatial interactions which invoke integrated visualizations. And an integrated interaction can invoke visualizations exclusively in the spatial domain as shown in the middle-right cell. Analogously, an integrated interaction can invoke a visualization change in the abstract domain only. A detailed description of visualizations and interactions in our integrated space can be found in Section 4. The numbers in the matrix cells in Figure 4 correspond to section numberings where each *cell* is discussed.

The matrix contains two empty cells. These represent abstract or spatial interactions that result in visualizations exclusively in the other domain. We do not provide examples of these types of interactions because an interaction in one domain will naturally lead to a visualization in the domain of its origin.

4 Integrating between Abstract and Spatial Domains

The following subsections describe the different techniques and approaches created to generate an integrated visualization of abstract and spatial data. We first describe interactions and visualizations that apply to one domain only. The rest of this section is dedicated to the description of the integrated visualization space.

4.1 Abstract Interaction and Abstract Visualization

The category of abstract interaction and abstract visualization is located in the top-left cell of Figure 4. This corresponds to visualization and interaction possibilities typical for graphs and trees in the information-visualization domain. The abstract data is rendered as a node-link diagram. We utilize standard graph layouts such as force-directed layouts and Balloon trees [11]. The nodes are rendered as circles with the name of the structure as a label on the top half of the circle. The color of the node can be changed to convey state-change information to the user. With the same intent in mind the edges between nodes can also be colored. Nodes can be focused, selected, or resized. Selecting a node other than the root makes the selected node the new root and removes all other nodes that are not part of the sub-tree below the selected node. In addition the path to the original root is included. Figure 1 shows the result of selecting the vertebrae as the new root. Removing specific sub-trees is possible by collapsing a node. Transitions

between interactions with the abstract data are animated. The interaction possibilities on the abstract data will be integrated with the spatial domain in the following subsections.

4.2 Spatial Interaction and Spatial Visualization

Spatial interaction and visualization is depicted in the bottom-right cell of Figure 4. This category corresponds to a straightforward visualization of the spatial data with typical interaction possibilities. We display the spatial data using volume rendering and slicing. The volume rendering is aware of segmentation data and individual visual styles can be applied to the different segmentations. In the spatial domain the viewpoint for volume rendering can be relocated, the visual style can be changed, the slicing plane can be moved along the three main axes, and the structure located under the mouse cursor can be identified.

4.3 Abstract Interaction and Integrated Visualization

This category of interaction and visualization consists of interactions typical for the abstract domain, such as node focusing, that leads to visualizations in both domains. This category corresponds to the top-center cell in Figure 4.

Colored edges and styled structures: Navigating the abstract space and focusing a node in the hierarchy results in the volumetric structure being automatically visually emphasized using a set of predefined styles and colors. To increase overview locally, the edges between nodes are also colored. The same colors applied to the volumetric structures are assigned to the edges. The edge between the node and its parent is colored in black. This is shown in Figure 6.

The reason for this technique to fall into this category is that the interaction is only with the tree layout, e.g, focusing a node. The result is visualized in both domains, i.e, styling of nodes, edges and volumetric structures.

Pruning: Volume rendering of structures that spatially enclose interior objects results in occluded features. Changing the visual representation of the occluding structures to transparent enables a clear view of otherwise occluded parts. The possibility to remove occluding structures has been realized through interactions with the graph. Typical interaction operations with trees are collapsing or pruning of sub-trees. For the graph display this means to remove from the layout all nodes included in the sub-tree. For volume rendering this means complete removal of the associated 3D structures. By collapsing a node, the sub-tree is effectively removed from the visualization in both visualization domains.

When a sub-tree is collapsed, the sub-tree root is replaced by a small node with a *plus* symbol. It enables a future expansion of the sub-tree. This interaction operation allows a user to create a specific, desired subset of the entire structure. For example, studying the cortex of the brain, it is possible to remove all of

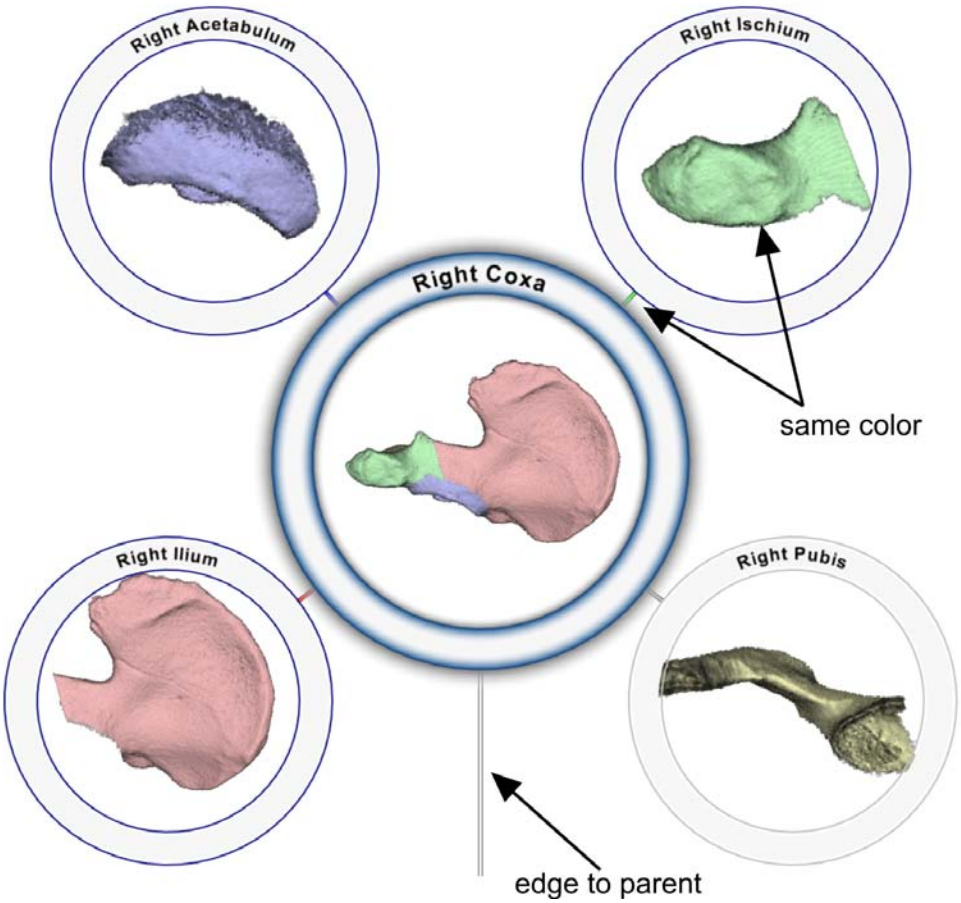


Figure 6: Colored edges with direct relation to structure color. The gray color of the pubis node indicates that this structure is not visible from the current viewpoint of the selected node.

the sub-cortical structures. An example of pruning is shown in Figure 7 where specific bones have been removed from the foot. This makes it easier to study the interface between bone segments and neighboring bones in context.

The interaction in this technique also applies only to the tree layout but results in visual changes in both domains. The sub-tree that was pruned is effectively removed from the display. This produces the side-effect of removing for all ancestral nodes the spatial structures associated with the pruned sub-tree.

4.4 Spatial Interaction and Integrated Visualization

Spatial interactions that influence integrated visualizations is the category located in the bottom center cell of Figure 4.

Picked-structure path: Picking is in the spatial visualization of complex

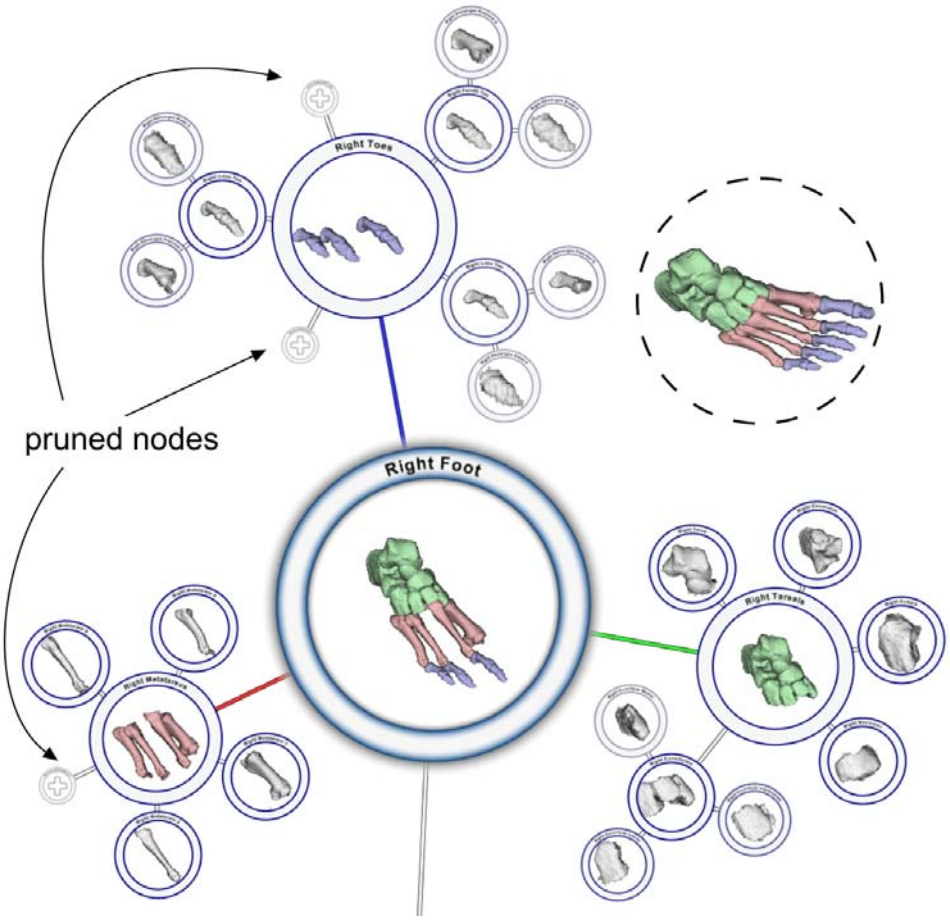


Figure 7: Pruning of big toe, middle toe and metatarsal. Collapsed nodes are shown as circled plus symbols. The dashed circle shows the foot before pruning.

volumetric structures a straightforward interaction for selecting a sub-structure. This operation is realized by casting a ray through the volume. When a particular structure is selected, visual prominence is given to this structure. To convey how this structure is positioned with respect to the abstract hierarchy, the corresponding graph node is emphasized to effectively indicate its hierarchical location. The structure is highlighted under the mouse cursor and the path from the focused node to the graph node representing just the picked structure is highlighted. Figure 8 shows a mouse pointer picking a specific structure and the structure is emphasized with an orange color in the volume rendering. The path to the structure itself, is highlighted with orange outlines on edges and nodes in the graph.

This is an integrated visualization since the nodes and edges that include the picked structure are highlighted while the picked structure in the spatial domain is highlighted as well. It is a spatial interaction because the structure is associated

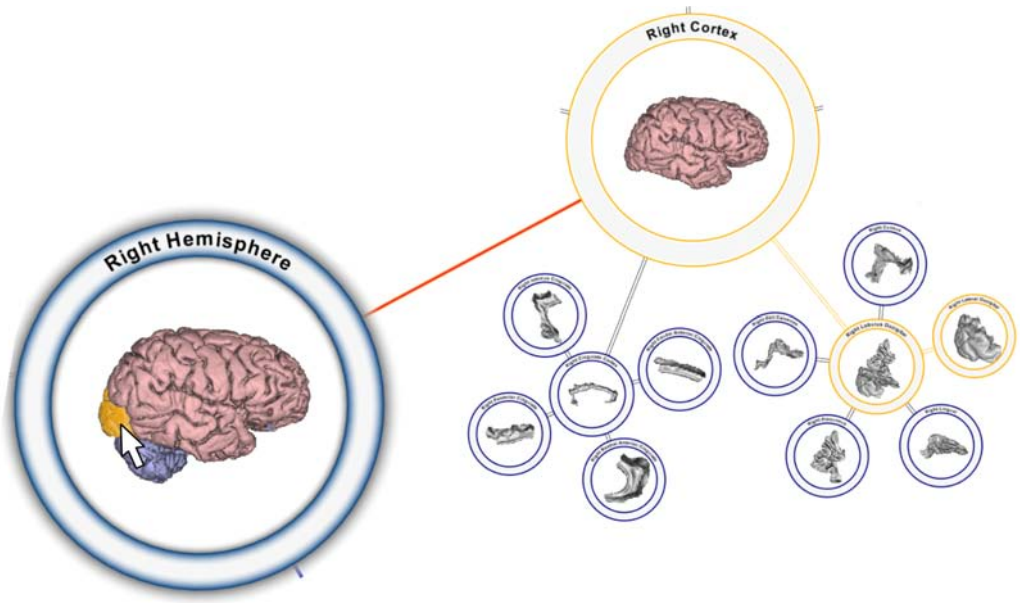


Figure 8: Picked-structure path with orange highlight on edges and nodes. The picked-structure is highlighted in orange in the selected node. The mouse cursor is exaggerated in size.

with a single segmentation and no hierarchy information is necessary to identify it.

Slice intersection: In a medical environment slicing is an often used technique of visualizing and interacting with volumetric data. A slicing interaction shows a cross-section through the structural information, and partitions the volume into two sub-volumes. Our integrated visualization represents this partitioning on the graph. The slicing plane's relative position to a structure is visualized through node coloring. The spatial extent of a structure is defined as the structure's minimum and maximum coordinates in the slicing direction. If a structure's maximum extent value is less than the slice position the node is colored green. This can be interpreted as the slicing plane being in front of the structure. If the slice position is less than the minimum extent of the structure, the node is colored red. This is interpreted as the slicing plane being behind the structure. When the slicing plane intersects the structure, i.e., the current slice position is between or equal to the structure extents, the node is colored blue. This visualization can be seen in Figure 13(a). It provides a useful and fast way of getting an overview on which structures are part of the current slice. The visual impact of this technique can be seen in Figure 9. Changing the zoom level from overview to focus, a later described technique (hierarchical liftcharts), provides much more detailed information about the relative positioning of the slice.

The interaction approach in this technique is changing the slice position and

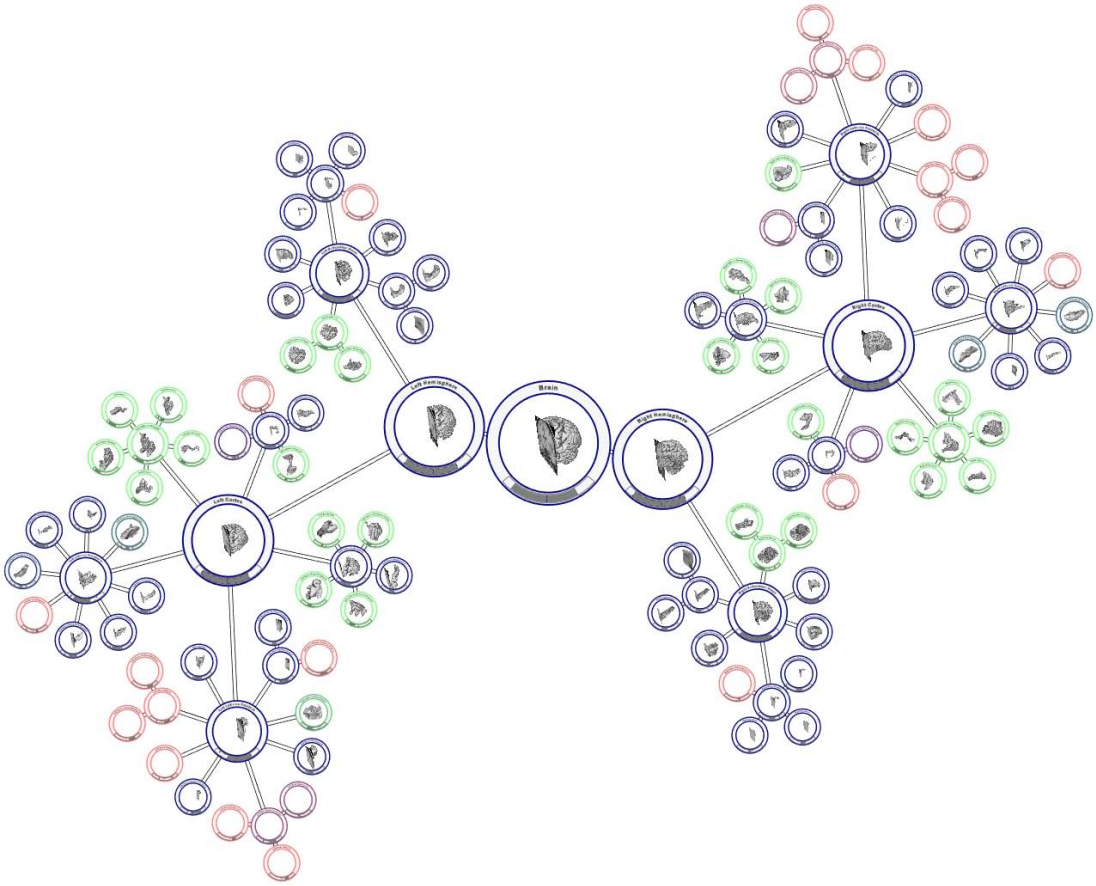


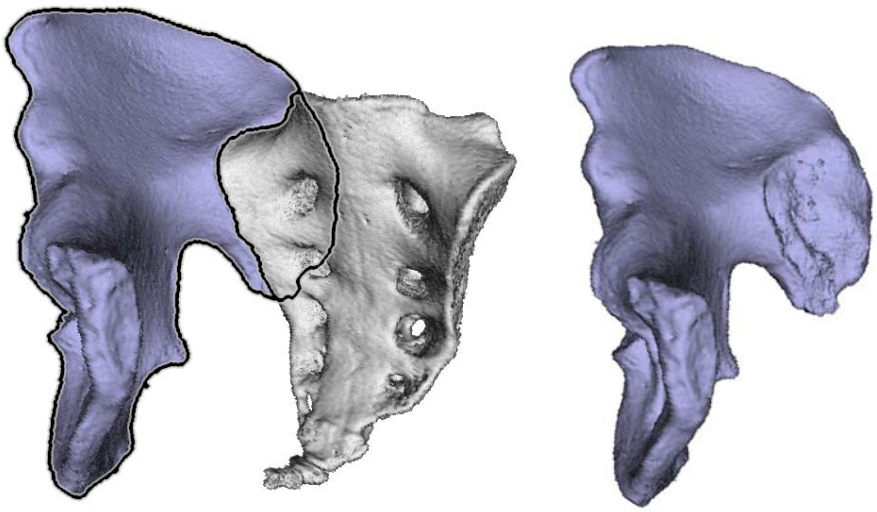
Figure 9: Visual impact of slicing. The node color indicates the relative position of the slicing plane with respect to the node structure. Green, red, blue means the slicing plane is in front, behind or intersecting the node structure.

is in the spatial domain only. Visualizing the result affects both domains. The slice is displayed together with the volume rendering and the node color changes based on the relative slice position.

4.5 Integrated Interaction and Spatial Visualization

This category of interactions and visualizations results in visual output only in the spatial domain. The middle-right cell in Figure 4 represents this category.

Selection outline: In the spatial domain a high level structure may be composed of several substructures that occlude each other and it is difficult to see where a specific hierarchical substructure is spatially located. To help the user to locate a selected feature and indicate which parts of the structure may be occluded, an outline of the structure is visualized. This interaction takes advantage of visual motion cues to better convey the shape of the analyzed structure. The



(a) (b)

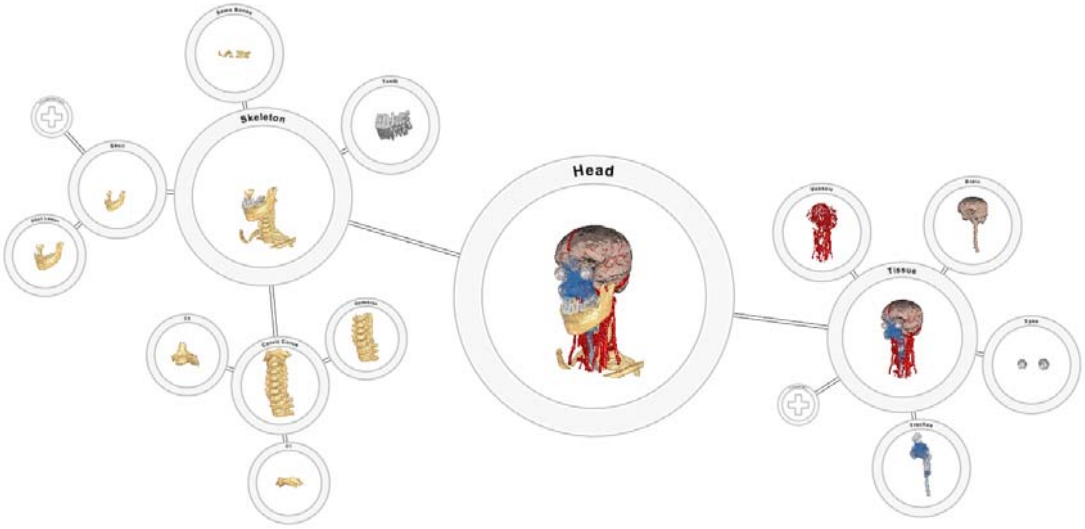
Figure 10: (a) Coxa occluded by the sacrum. The outline indicates the extent of hidden structures. (b) Coxa with no occlusion.

outline is applied to the whole structure and also indicates the border between the visible part and the occluded part. This is shown in Figure 10(a). In Figure 10(b) the occluded parts of the Coxa have been revealed.

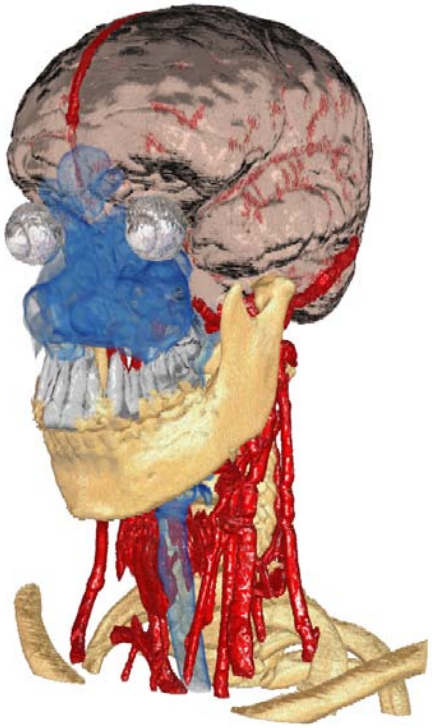
This technique is an integrated interaction because it relies on hierarchical and spatial information to identify the structure to outline. A list of segmentations that are part of a hierarchical structure is used to identify the voxels that are part of the outline. The resulting visualization applies to the volume rendering only.

Hierarchical visual style: Taking advantage of the information in the abstract space creates an intuitive way to change the visual representation of structures in the spatial domain. Changing the visual style of a higher level structure, results in the new style being propagated down in the hierarchy to all lower level features. This results in increased efficiency to refine the visual appearance of the visualized structures. For example, it is possible to first select a visual representation that displays all structures in the same color and then refine for substructures. In Figure 11 this is illustrated by changing the style of all dense tissues to a bone-like visual representation. Afterwards, the remaining soft tissue structures are refined by individually changing their color and style. This approach is increasingly efficient for larger hierarchies.

It is again an integrated interaction. Changing and applying the style is a typical interaction metaphor in the spatial domain. For the style applied to a structure to propagate to all child nodes information about the hierarchy is necessary. The resulting visualization only applies to the volume rendering of the



(a)



(b)

Figure 11: (a) Different style transfer functions applied to structures. Upper skull (left $\oplus$) and skin (right $\oplus$) are removed to expose inner structures. (b) Close-up at root node.

structures.

4.6 Integrated Interaction and Integrated Visualization

In the most general case integrated interactions are performed in both domains to invoke a visualization and the invoked visualization is applied to both domains simultaneously. This category is located in the central cell in Figure 4.

Occluded structures: Manipulation of the viewpoint is a frequently used interaction in 3D with structural volumetric information. A chosen viewpoint also determines which structures are visible and which are occluded. This information can be extended to the hierarchical visualization by color coding those nodes and edges which are visible from a particular viewpoint and which are occluded.

Looking at a hierarchical structure that is composed of several substructures, one or more of the substructures may be occluded. In this situation we indicate to the user which of the substructures cannot be seen from the given viewpoint. Visibility is defined as the ratio between the number of pixels rendered for a substructure and the total number of pixels for the complete structure. If a structure is completely occluded, this is conveyed to the user by changing the color of the node. The color of the node is gray when the structure is less than 1% visible. Otherwise if the structure is less than 5% visible the color is interpolated between gray and blue. If the visibility is 5% or more then the node is rendered in blue. In Figure 12 this effect is demonstrated on an overview of the brain. The left hemisphere is completely occluded by the right hemisphere. This is easily perceivable as all nodes on the left part of the image are shown in gray. Some structures in the right hemisphere are also not visible from this viewpoint. Thus some nodes on the right part of the image are gray as well. Another example can be seen in Figure 6 where the right pubis, the yellow structure, is occluded by the rest of the coxa.

The spatial interaction for this technique is changing the viewpoint through a rotation and the abstract interaction is focusing on a node of interest. The focused node is used to determine which segmentations to check for occlusion at all levels of the hierarchy. The resulting visualization is using the new viewpoint for the spatial data while indicating the level of occlusion with color on the node outline.

Hierarchical liftcharts: In addition to showing the slice plane, we provide additional information about the structures on the node representation. In the bottom half of the node we render a so called slice bar that represents the full extent of the entire volume in the slicing direction. It is labeled with (1) in Figure 13(b) and 13(c). In these figures the extent of the structure represented by the node is shown as a gray ring sector labeled with (3) and the extent of the parent structure is labeled with (2). The current slice position is rendered as a red line labeled with (4). If the node is selected, the extents of all child structures

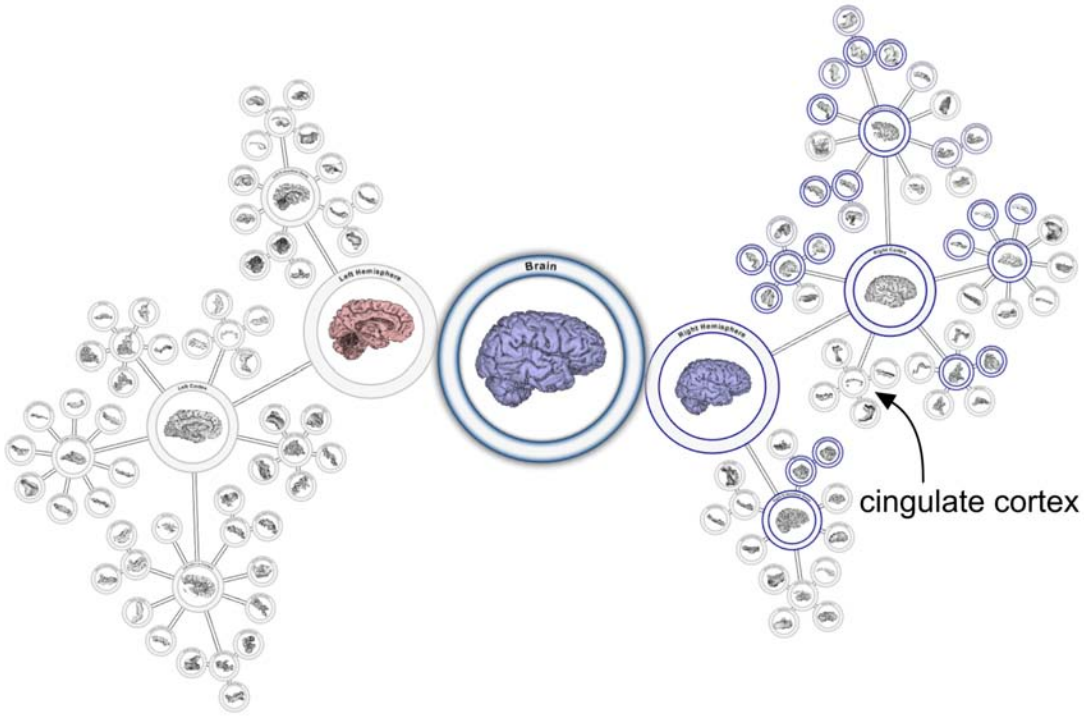


Figure 12: Gray nodes indicate occluded structures. The left hemisphere and several structures in the right hemisphere are occluded.

are indicated in the slice bar using the same colors as for the volume rendering and for the edges. The extents are labeled as (5) in Figure 13(c).

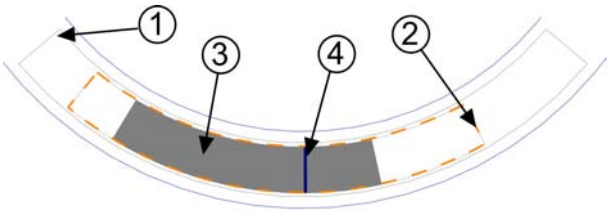
Hierarchical liftcharts are integrated interactions because they require information about the current slice position and also whether the node is focused or not. A focused node results in a different visualization than an unfocused node. The visualization consists of rendering the current slice and the slice bar which is depicted in the bottom part of the node.

4.7 Integrated Interaction and Abstract Visualization

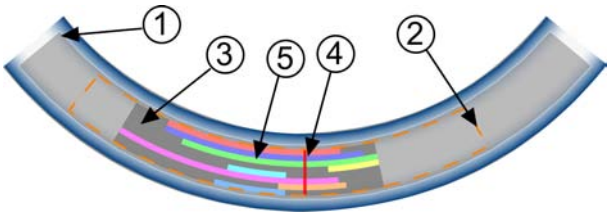
The middle-left cell in Figure 4 represents the category of integrated interactions that result in abstract visualizations. This category of interaction and visualization uses hierarchical and spatial data but provides visualizations that only applies to the abstract domain.

Property labeling: Let us assume that a whole series of data sets is available, e.g., from a longitudinal study. It might be interesting to see how a specific dataset deviates from the average of the series. Figure 14(a) illustrates our approach in this respect. The structure sizes (voxel counts) for several segmentations in the brain have been measured in a certain population. We compare the

(a)

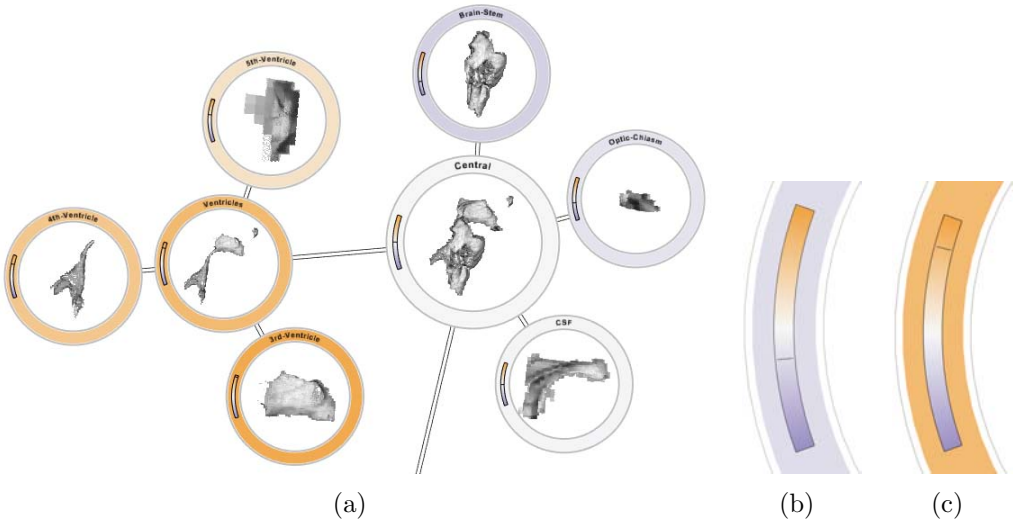


(b)



(c)

Figure 13: (a) Node rendering with selected node. A green node indicates the slicing plane is in front of the structure, a red node indicates the slicing plane is behind the structure and a blue node indicates a slicing plane that intersects the structure. Closeup of (b) unselected slice bar and (c) selected slice bar. In (b) and (c) the bar shows the bounding volume (1), parent extent (2), structure extent (3), slice position (4) and extent of children structures (5).



(a) (b) (c)

Figure 14: (a) Several nodes colored based on the deviation from an average structure. Orange is above, purple is below and white equals the average. (b) Legend indicating structure is below average. (c) Legend indicating structure is above average.

visualized brain against the average of the series.

Figure 14(a) shows a part of the brain and gives a comparison with the average structure size. The distance from the average is indicated in color. The color scale is from orange to purple, where orange encodes an above average situation while purple encodes a below average situation. The relative deviation can also be read off from the deviation legend shown in Figures 14(b) and 14(c). The legend shows the color scale and with a black line the location of the color applied to the node. The deviations from the average are aggregated hierarchically. The average is calculated for all structures and compared hierarchically.

The hierarchical aggregation of a chosen statistics requires both abstract data and spatial data. The resulting visualization is a color change in the abstract domain only.

Scatterplots: In a longitudinal study it is common to also record more than one metric. To visualize such information, scatter plots have been included inside the node rendering. This can be seen in Figure 15(a). A close-up of a scatter plot is shown in Figure 15(b). The scatter plot shows the relationship between a patient's age and the number of voxels for a structure. Blue dots represent males, pink dots represent females and the green dot is the current patient. The diagonal lines are separate linear regression lines for each sex. The scatter plots are, similarly to the property labeling, aggregated hierarchically.

The hierarchical aggregation of a chosen statistics requires both abstract data and spatial data. The resulting visualization is a new abstract visualization of

the statistical data composited on top of the volume renderings.

5 Implementation

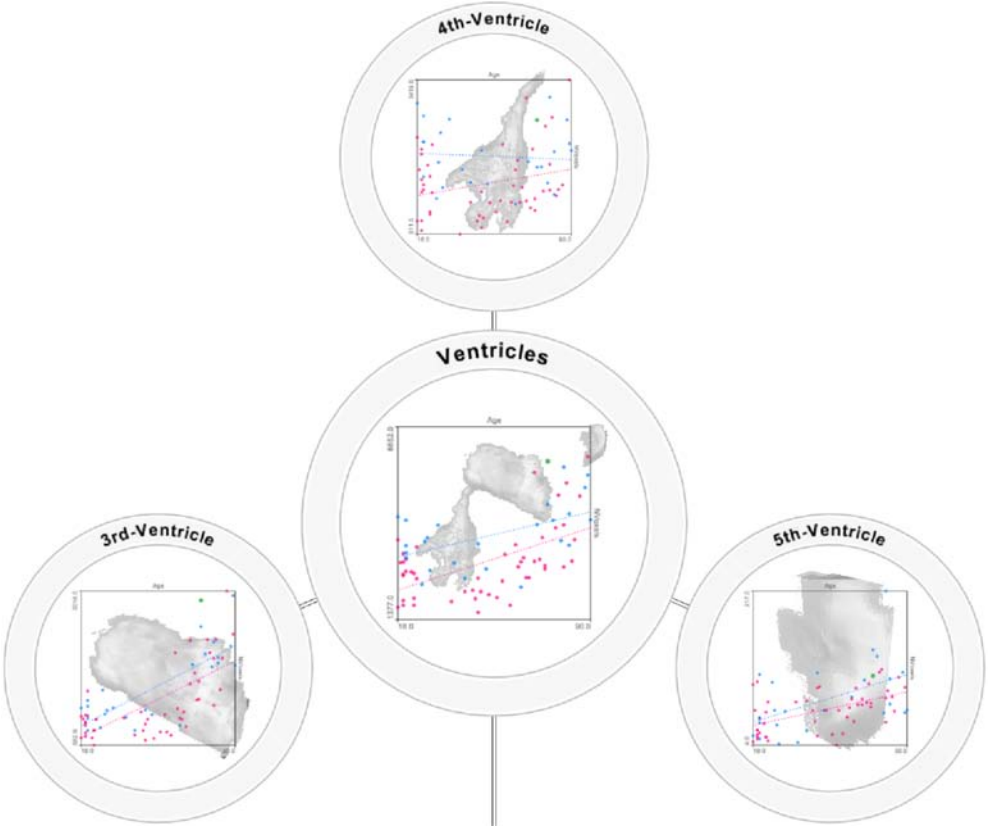
The implementation of the rendering system has been done in Java. OpenGL and the OpenGL Shading Language have been used for the graphical rendering. An off-the-shelf graph layouting library is used to position the nodes according to the Balloon placement algorithm. Rendering the node tree has been implemented as a multi-pass algorithm using the visitor pattern [6]. Every pass renders one layer of the final image and the layers are composited together.

The algorithm calculating the selection outline described in Section 4.5 and shown in Figures 1 and 10(a) is a pixel based approach for finding edges of structures. In a separate raycasting pass over ancestral structures of a focused node, a buffer is filled with values that represent one of three cases: 0 if the ray does not hit any segmentations associated with the focused structure, 1 if the ray hits a focused segmentation and 2 if the first segmentation hit is of the focused structure. The resulting buffer is then processed to identify two types of edges by checking gradients: from segmentation hit (1 and 2) to no segmentation hit (0) and first segmentation hit (2) to segmentation hit (1). The identified edges are colored in black. The resulting lines are then dilated to increase thickness and a halo is added to increase visibility. Finally the outline is overlaid on top of the volume raycasting image.

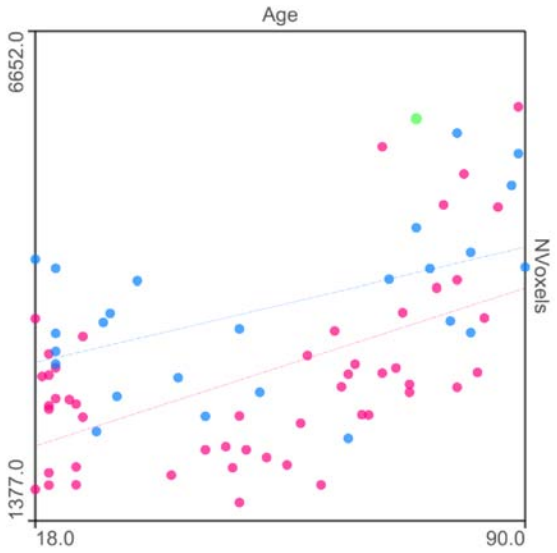
The occluded-structures algorithm described in Section 4.6 and shown in Figures 6 and 12 assumes that structures are opaque. In the same pass when calculating the selection outline the identity of the first segmentation hit is stored. The number of pixels for each segmentation is counted using the OpenGL extension ARB_occlusion_query. The total number of pixels for a hierarchical structure is summed up and the relative size of the substructure is calculated. In Figure 6 the focused structure is the right coxa. Since the number of pixels from the pubis that contribute to the image is zero, the pubis from the point of view of the coxa is completely occluded.

6 Results

The following results are demonstrated on several different datasets. Figures 1 and 11 use a CT scan of the head and neck with contrast enhanced sinus veins, at a resolution of $512 \times 512 \times 333$, with several anatomical structures segmented. Figures 6, 7 and 10 are generated with a segmentation of the right leg of the Visible Male CT dataset [12] in full resolution cropped to a resolution of $268 \times 243 \times 1136$. Figures 8, 9, 12 and 13 use the Bert dataset provided by FreeSurfer [5]. Finally, the dataset used in Figure 14 and Figure 15 are from the OASIS brains



(a)



(b)

Figure 15: (a) Shows several nodes with a composite of volume rendering and scatter plots. (b) A closeup of the ventricles scatter plot from (a).

database [13] consisting of more than 400 segmented brains with associated meta information, such as age, gender, education and so on. Both Bert and OASIS volumes have a resolution of $256 \times 256 \times 256$.

In Figure 1 the spatial relationships between the hierarchically grouped structures are visualized for the cervical curve as outlines in the ancestral nodes. The cervical curve itself is composed of three child structures and the relative position and shape of these are easily discernible from the focused node. It is also easy to see that the structure is not visible from the viewpoint of the head node, where the vertebrae are covered by several layers of tissue. This is encoded by rendering the outline of the head node in gray.

The selection outline to indicate the spatial position of a structure is also used to highlight areas of a focused structure that are occluded by surrounding structures. In the skeleton node of Figure 1 it is possible to see that parts of the jaw and bones of the shoulder partially occlude the cervical curve. Figure 10(a) presents this in more detail, where it is quite obvious that the coxa is partially occluded by another structure. In addition this visualization indicates the shape of the occluded structure and provides a better insight into the structural relationships.

Similar to the occluded structure in the head in Figure 1 the pubis of the coxa in Figure 6 is not visible from the selected viewpoint. Working with a larger hierarchy the impact of such a visualization is evident in Figure 12 where the viewpoint is chosen to completely hide the left hemisphere. Only nodes with a blue outline are visible from this particular viewpoint.

Figure 11(a) is an example of a user-generated structure where substructures have been removed to reveal objects of interest. In this example the location of the eyes and the trachea relative to the brain is of interest so the upper skull and the skin have been removed. The same approach has been used in Figure 7.

Figure 8 helps the user in identifying an object of interest. The structure under the mouse cursor is highlighted and in addition the hierarchical path to the structure is shown.

We have given slicing a semantic meaning in the abstract domain which is illustrated in Figure 9. In this overview it is easy to see which structures are intersecting the slice plane. When zooming in to see the details of a specific structure, more information about the slicing is available. In Figure 13(a) the current slice position is shown in relation to parent and child structures.

The OASIS brains database contains a longitudinal study of a certain population. All the brains have been segmented and one of the metrics that is interesting to study is the relative size of structures. Figure 14(a) shows a fast way of identifying structures that deviate from the average. It is, for example, very easy to see that the ventricles (orange nodes on the left) are much larger than the average.

On a Dell Precision T5400 using a single thread with NVidia 280 GTX the system performs at interactive speeds. For example, rendering the image seen in

Figure 12 at a 1000×1000 pixel resolution we achieve a performance of approximately 11 fps. Zooming in to only render the sub-tree shown in Figure 13(a) increases the performance to 40 fps. The increase is mostly due to the reduced number of visible structures. The largest performance bottleneck of the system is the volume resolution. Larger volumes increase the processing times of the raycasting, selection-outline and occlusion-testing algorithms.

7 Conclusions and Future Work

In this paper we have introduced an integrated visualization that bridges spatial and hierarchical domains. We have proposed a classification of visualizations and interactions that can be organized in a 3×3 matrix depending on whether they are of abstract, integrated, or spatial nature.

The increased occurrence of quite heterogeneous data sources for the same real-world phenomenon requires integrated visualization approaches. The currently available algorithms are mostly tailored to a specific data space, e.g., abstract or spatial. In this sense the algorithms are scattered points in a design space for visualization algorithms. The increasingly prevalent heterogeneous data sources make it necessary to develop new algorithms. One way to do so, is to perform *"scattered data interpolation"* in the aforementioned algorithm design space. The approach in this paper is one example where we *"interpolated"* between graph drawing and volume rendering. We believe that many more algorithms can be discovered by further *"reconstructions"* in this design space.

In the case of integrated visualizations invoked by interactions in one or the other domain, a useful approach in developing new techniques is realized as a two-stage process. First we identify a fundamental interaction metaphor in one domain, such as *select* or *show*, for example. Then we seek for a specific visualization in the other domain that realizes the respective meaning of this interaction. This way an interaction can result in visualizations that expand beyond the borders of its origin domain. Examples of this type can be found in Sections 4.3 and 4.4. For integrated interactions we have not found such a systematic integration approach and their discovery was rather stimulated by practical needs and experiments.

Illustrative visualization technology covers many techniques that mimic the approaches that illustrators use. These techniques include exploded views, cutaway views, peel-aways, labels and many other techniques to achieve visualizations with an illustrative presentation. Illustrations that present a scientific topic often have to convey hierarchical information, but only a few of the mentioned techniques are directly applicable in the hierarchical context. We see our work as one element of interactive direct volume illustrations which specifically addresses the hierarchical aspect of scientific data. With this in mind it would be interest-

ing to take the idea of interactive illustrations one step further and to combine several of these illustrative techniques in establishing a visualization toolbox for interactive poster generation. Our work will be the tool to show the hierarchical characteristics within the 3D structures.

8 Acknowledgments

We would like to thank Stefan Bruckner, Daniel Patel, Peter Rautek, Maurice Termeer and Martin Haidacher for helpful suggestions and comments.

References

[1] J.-P. Balabanian, M. Ystad, I. Viola, A. Lundervold, H. Hauser, and M. E. Gröller. Hierarchical volume visualization of brain anatomy. In *VMV 2008, Vision, Modeling and Visualization*, pages 313–322, Oct. 2008.

[2] S. Bruckner and M. E. Gröller. Style transfer functions for illustrative volume rendering. *CGF*, 26(3):715–724, Sep 2007.

[3] M.-Y. Chan, H. Qu, K.-K. Chung, W.-H. Mak, and Y. Wu. Relation-aware volume exploration pipeline. *IEEE TVCG*, 14(6):1683–1690, 2008.

[4] K. Engel, M. Hadwiger, J. M. Kniss, C. Rezk-Salama, and D. Weiskopf. *Real-time Volume Graphics*. A. K. Peters, 2006.

[5] Freesurfer. http://surfer.nmr.mgh.harvard.edu, 2009.

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[7] H. Hauser. *Scientific Visualization: The Visual Extraction of Knowledge from Data*, chapter Generalizing Focus+Context Visualization, pages 305–327. Springer, 2005.

[8] I. Herman, G. Melancon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE TVCG*, 6(1):24–43, 2000.

[9] J. Krüger and R. Westermann. Acceleration techniques for GPU-based volume rendering. In *IEEE Visualization*, pages 287–292, 2003.

[10] W. Li, M. Agrawala, B. Curless, and D. Salesin. Automated generation of interactive 3d exploded view diagrams. *ACM Trans. Graph.*, 27(3):1–7, 2008.

[11] G. Melancon and I. Herman. Circular drawings of rooted trees. Technical Report INS-R9817, CWI, Amsterdam, Netherlands, 1998.

[12] The visible human project. http://www.nlm.nih.gov/research/visible/, 2009.

[13] Open access series of imaging studies (OASIS). http://www.oasis-brains.org/, 2009.

[14] A. Pommert, Schubert, Riemer, Schiemann, Tiede, and Höhne. Symbolic modeling of human anatomy for visualization and simulation. In *Vis. in Biomed. Comp.*, volume 2359, pages 412–423. SPIE, 1994.

[15] C. Tietjen, B. Meyer, S. Schlechtweg, B. Preim, I. Hertel, and G. Strauß. Enhancing Slice-based Visualizations of Medical Volume Data. In *Eurographics / IEEE-VGTC Symposium on Visualization 2006*, pages 123–130. Eurographics, 2006.

[16] Y. Wang, S. Teoh, and K.-L. Ma. Evaluating the effectiveness of tree visualization systems for knowledge discovery. In *Eurographics / IEEE-VGTC Symposium on Visualization 2006*, pages 67–74, 2006.

Paper V

$\mathcal{A}$

Jean-Paul Balabanian* Eduard Gröller*†

Abstract

This paper describes the concept of $\mathcal{A}$-space. $\mathcal{A}$-space is the space where visualization algorithms reside. Every visualization algorithm is a unique point in $\mathcal{A}$-space. Integrated visualizations can be interpreted as an interpolation between known algorithms. The void between algorithms can be considered as a visualization opportunity where a new point in $\mathcal{A}$-space can be reconstructed and new integrated visualizations be created.

1 Introduction

Illustrative visualization has been quite successful in recent years. The idea of illustrative visualization is to mimic the traditional illustrators' styles and procedures. Many techniques have been developed that span a wide range of traditional styles. These techniques include lighting models that resemble illustrative styles, exploded views, labeling, ghosting, and halos and have been successful at simulating the original illustrators' results. One strategy of illustrators is in principle to blend together very different styles. For example in one part of an illustration a realistic representation of the object is shown while in another part of the drawing the object is shown using ghosting effects, halos or outlines. Figure 1 demonstrates this heterogenous blending of different styles with several examples of a car.

This approach is similar to what illustrative visualization is doing and the idea of *blending different styles* is a concept that can be transferred, in a metaphorical

*Department of Informatics, University of Bergen, Norway.

†Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria

Figure 1: An illustration of a car with different parts ghosted. Copyright beaudaniels.com

way, to *blending of different algorithms*. Merging the results from one visualization with the results from another visualization, in a non-trivial way, can be considered as blending between the two algorithms. A simple example of this concept can be derived from slicing and volume rendering. These are two different visualization techniques for volume data. Blending between the two techniques could result in a visualization where the slice is integrated into the volume rendering. Figure 2 shows an example of what an integrated visualization combining direct volume rendering and slicing may look like.

Integrated visualizations solve a limitation typical for linked views which comes from an increased data complexity. In a linked-views setup the number of views increases with the complexity of the data. As the complexity increases the number of interesting aspects of the data also increases and more views are necessary to convey all of the important parts. Integrated visualizations alleviate this problem by providing a single frame of reference for all visualizations.

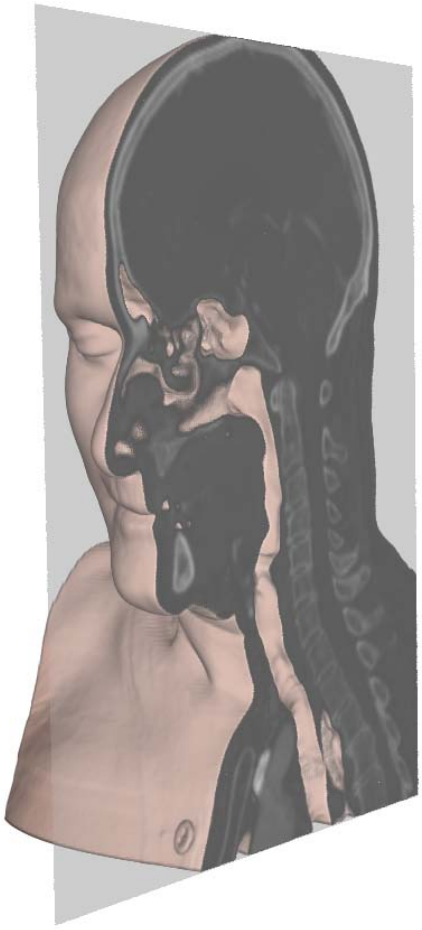


Figure 2: A volume-rendering with an integrated sagittal slice.

They incorporate all of the important aspects of the data into the same view. Creating an integrated visualization is not straightforward, though, and so far only rather ad hoc approaches to integrated views are known. A more systematic approach to create integrated views might be $\mathcal{A}$*-space*. $\mathcal{A}$-space is a space where all visualization algorithms reside. In $\mathcal{A}$-space every algorithm is represented by a unique point. $\mathcal{A}$-space is sparsely covered by the known visualization algorithms and there are many voids. Filling the voids between the points leads to reconstruction in $\mathcal{A}$-space and new integrated visualizations.

In the next section we will give examples of different visualizations that are reconstructions in $\mathcal{A}$-space and we will describe what type of integration each example is. In Section 3 we will describe $\mathcal{A}$-space in more detail and we will conclude in Section 4.

2 Examples of Reconstruction in $\mathcal{A}$-Space

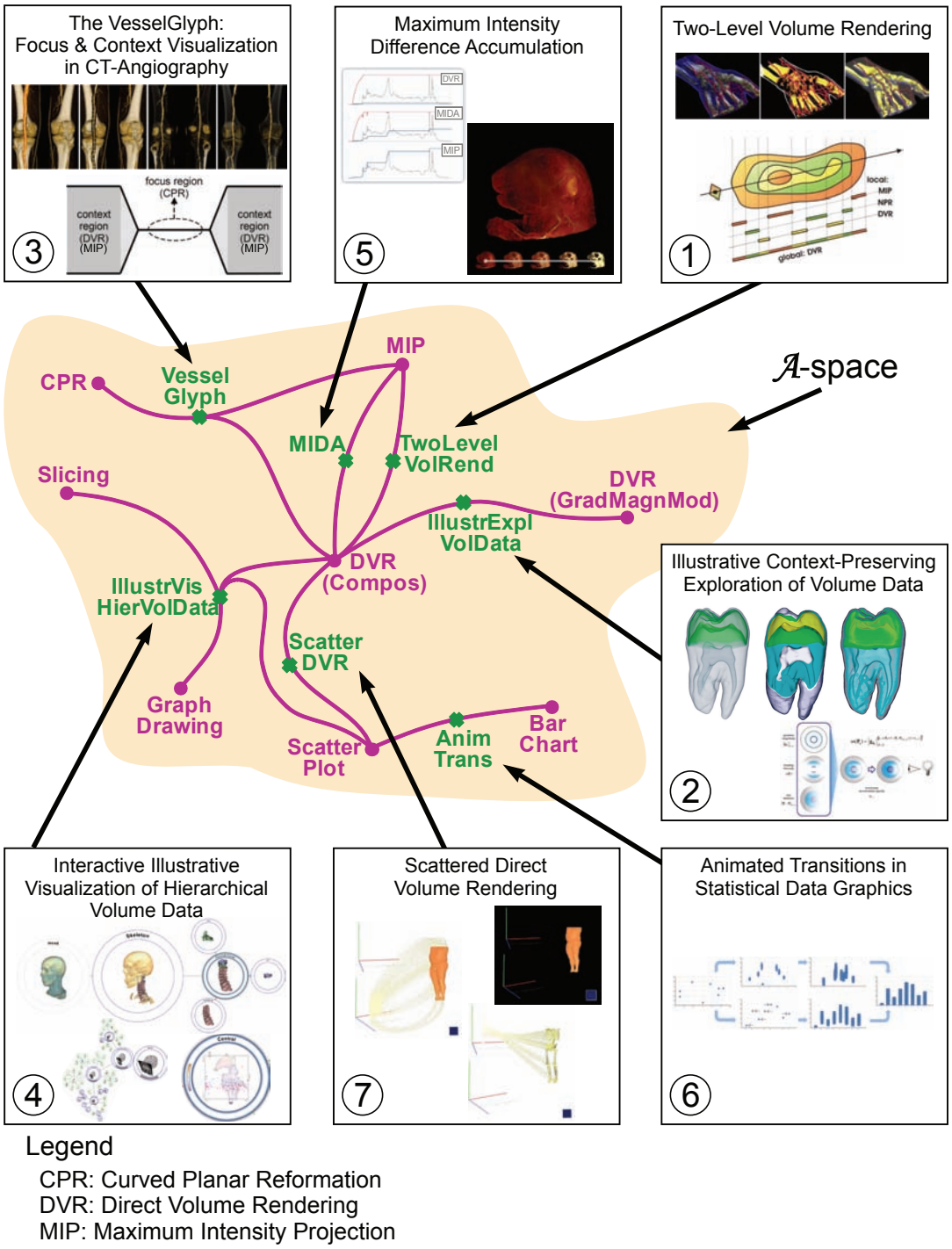
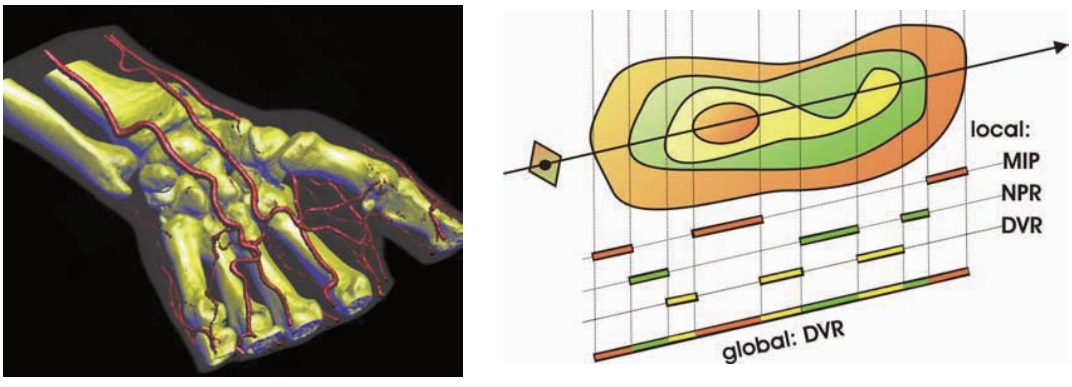


Figure 3: $\mathcal{A}$-space with example population.



(a) (b)

Figure 4: (a) A result from the Two Level Volume Rendering [5, 4] visualization technique. One specific technique is used for the bone, another one for the skin and a third one for the vessels. (b) A schematic of the algorithm selection during rendering (NPR: Non-Photorealistic Rendering).

In this section we will showcase several examples of visualizations that are a *blending of algorithms*. The integrated visualizations presented can be considered as reconstructions in $\mathcal{A}$-space with known points as origins. We will show where in $\mathcal{A}$-space these visualizations reside and the component algorithms required to perform the reconstructions. Figure 3 is a schematic of $\mathcal{A}$-space with several algorithms indicated. The pink points represent well known algorithms that in principle are not integrated visualizations. Between these points paths have been drawn with green crosses indicating the reconstructed algorithms. An interesting observation is that between MIP and DVR there are two different paths. A path represents one way of blending algorithms. In $\mathcal{A}$-space several paths may exist between algorithms and may result in fundamentally different visualizations. In the following sections we will describe all of the example visualizations that are present in Figure 3. We will discuss them in the order indicated by the number in the lower left of each frame. The chosen examples are just a subjective selection to illustrate a few nice places in $\mathcal{A}$-space. A comprehensive overview on previous integrated views is beyond the scope of this paper.

2.1 Two-Level Volume Rendering

The Two-Level Volume Rendering approach proposed by Hauser and Hadwiger [5, 4] is a merger of several visualization techniques. The idea behind the approach is to use different rendering techniques depending on the underlying data. The techniques available for the rendering are Maximum Intensity Projection (MIP), Direct Volume Rendering (DVR) and others. During volume rendering the appropriate visualization techniques for the underlying segmented regions are cho-

sen. The integration is a spatially coarse one since there is no smooth transition between the techniques and the resulting pixel is a composite of the visual representations produced by the different techniques. Figure 4(a) shows an example of this visualization approach where one technique is used for the bone, another technique for the vessels and a third technique for the skin. Figure 4(b) indicates that for different spatial regions different algorithms are employed.

2.2 Illustrative Context-Preserving Exploration of Volume Data

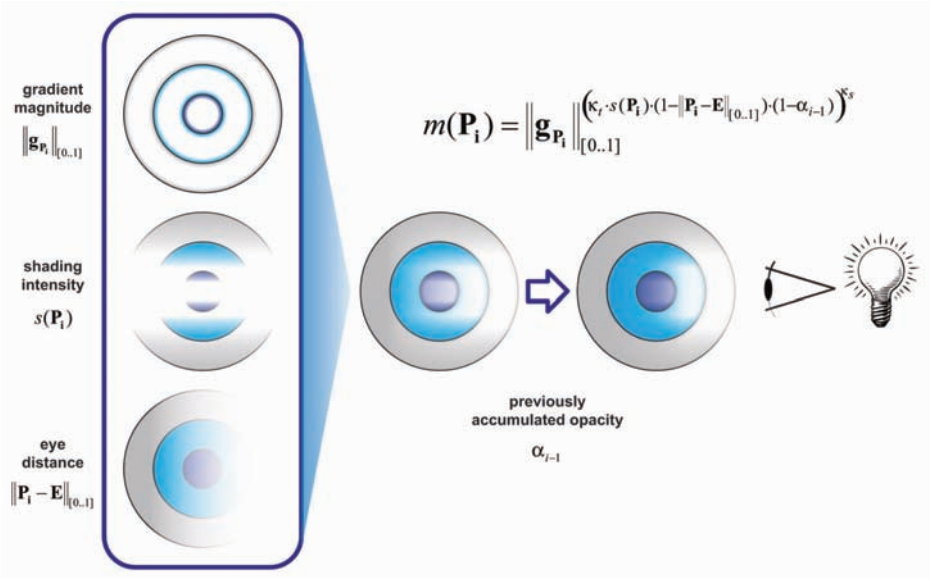
The Illustrative Context-Preserving Exploration of Volume Data technique proposed by Bruckner et al. [2] is a visualization technique that enhances interior structures during volume rendering while still preserving the context. Usually during volume rendering several structures may occlude the one of interest. Many techniques exist that can help in reducing the occlusion. Reducing the opacity of the occluding structures or applying clipping are two such techniques. The problem with these techniques is that they might remove the context of the interesting feature. The proposed approach combines DVR based on compositing with Gradient Magnitude Modulated DVR to reduce the opacity of less interesting areas in a selective manner. Two parameters are used to decide how to continuously *interpolate* between the two algorithms based on the input data. The results are illustrative volume renderings where contextual structures are outlined and the focused structures are kept in a prominent way. Figure 5(a) shows an example image produced by this technique. The center of the hand is semi transparent showing, among other details, the blood vessels quite prominently. The edges of the hand are not ghosted and thus retain the context. As indicated in Figure 5(b) the technique is a smooth and seamless integration of gradient based opacity modulation, selective occlusion removal, fuzzy clipping planes and multiple transparent layers' handling.

2.3 The VesselGlyph: Focus & Context Visualization in CT-Angiography

The two previous examples have shown integration between techniques that all operate in 3D. The following example is an integration between a 2D technique and a 3D technique. The result is a visualization that exploits the complementary strengths and avoids the complementary weaknesses of both.

The VesselGlyph, proposed by Straka et al. [8], is a technique that combines Curved Planar Reformation (CPR) with DVR. CPR is a technique that takes a feature like a blood vessel and cuts it with a curved surface revealing the inside structures. The shape of the surface is adapted so that it follows the curving and twisting of the structure. The resulting visualization is a 2D slice of the inside of

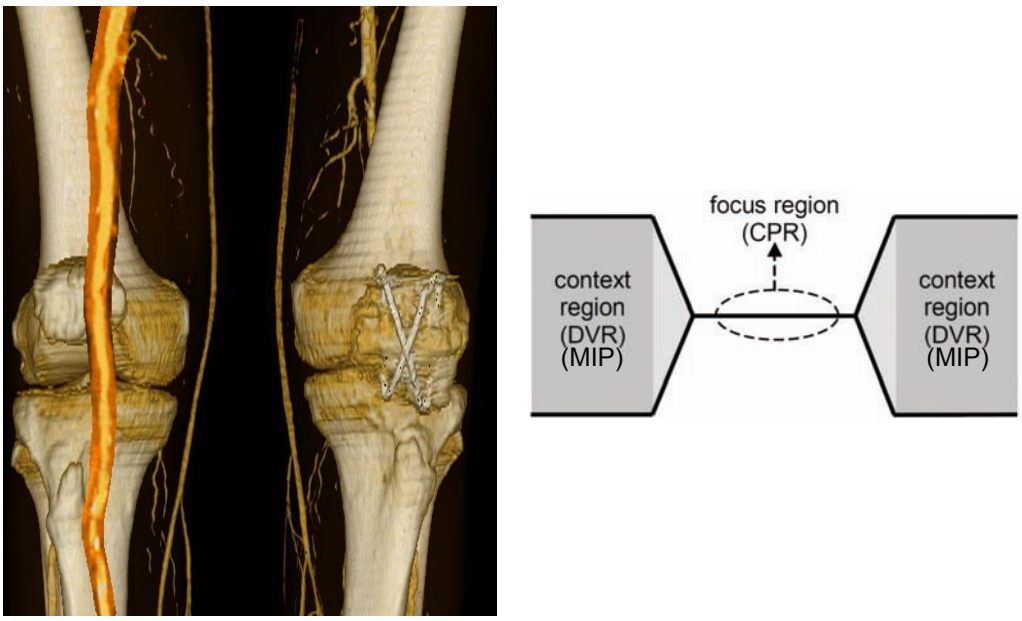
(a)



(b)

Figure 5: The context-preserving volume rendering of a CT-scanned hand produces similarities to the ghosting effect used by illustrators.

the vessel. The VesselGlyph technique incorporates the CPR slice into the DVR visualization of the context structures. The resulting visualization shows interior details of blood vessels with CPR presented in the correct context rendered with DVR. The type of integration employed in this visualization is the merging of two spatially registered visualizations using for example image compositing techniques. Figure 6(a) shows an example of this type of visualization. With DVR



(a) (b)

Figure 6: (a)The VesseGlyph [8] in action. The CPR, the vertical orange and red band on the left side, is projected onto the DVR of the same structure. (b) The concept of the VesselGlyph where CPR, the focus region, is integrated smoothly into the DVR, the context region.

alone the interior calcifications of blood vessels would not show up appropriately. With CPR alone the context region would be sliced arbitrarily which greatly reduces overview. Figure 6(b) depicts the concept of the VesselGlyph within an axial slice where the blood vessel would show up as a circular region in the slice center. CPR is considered for the focus region and is smoothly integrated into the DVR which is considered for the context region. It is also indicated that the context could alternatively be visualized using MIP.

2.4 Interactive Illustrative Visualization of Hierarchical Volume Data

The following example is much more complex than the previous ones. The visualization performs integration between 3D and 2D techniques and also between scientific visualization and information visualization techniques. The result is a visualization that in $\mathcal{A}$-space blends more than two different algorithms.

Hierarchical Visualization of Volume Data by Balabanian et al. [1] is an integrated visualization that uses graph drawing to visualize the hierarchical nature of structures in a volumetric dataset. Graph drawing in 2D is used as a guiding space where other 2D or 3D visualizations are embedded. The nodes in the graph

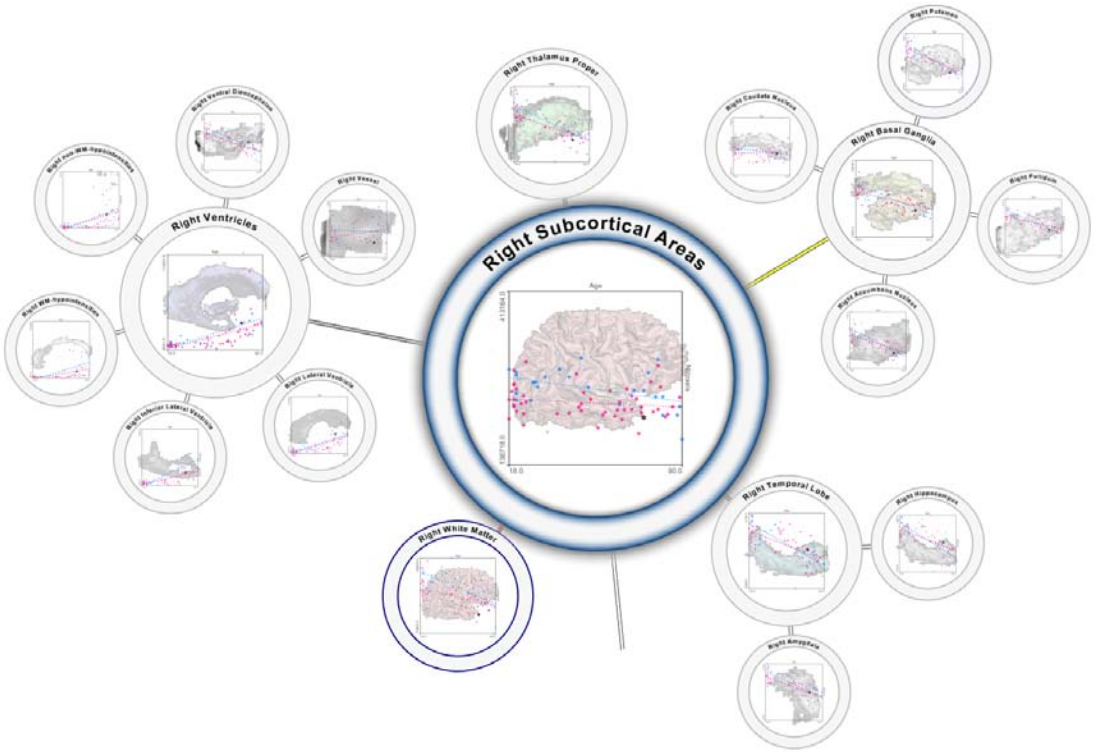


Figure 7: Hierarchical rendering of volume data [1]. A node-link diagram represents the hierarchical structuring with embedded volume renderings and scatter plots.

drawing are enlarged and serve as a canvas for the other visualizations. These visualizations include DVR, slicing, and scatter plots and are all integrated into one space. The type of integration employed in this visualization is at different levels. DVR and slicing are integrated in the same way as shown in Figure 2. The object that is actually rendered is defined by the hierarchical structure visualized by the graph drawing. The graph drawing is specified by the hierarchy information and every node is rendered as a circle. With statistical data available for the structures a scatter plot is added. Figure 7 shows an example of the subcortical areas of the brain (slicing not included here). The hierarchy of the substructures is visible with semi-transparent scatter plots on top of the embedded volume renderings.

In this example the integration is steered by the graph drawing. The abstract data is used to create a structure to present both the abstract and spatial data. It is also possible to envision an approach that uses the scientific visualization space as the embedding space. In Figure 8 we have sketched the imaginary interpolation between the two spaces that are part of the visualization, i.e., the abstract and the spatial space. The red circle indicates where this work is located but using

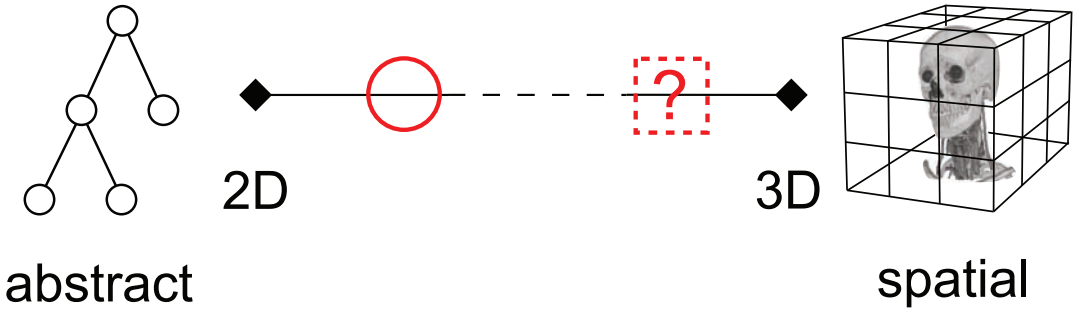


Figure 8: The red circle indicates the location where the interpolation between spaces takes place in the work by Balabanian et al. [1] while the dashed square indicates an alternative approach to this work.

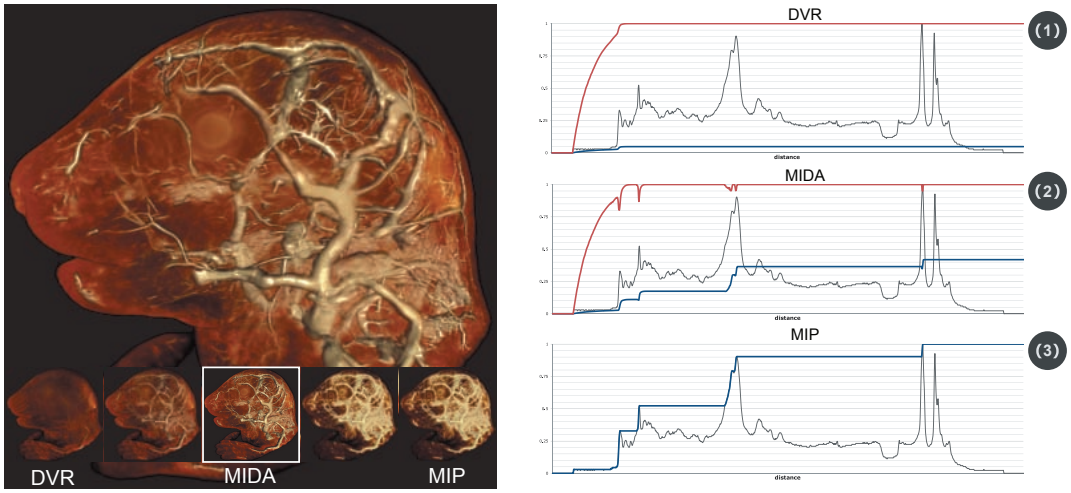
scientific visualization as the embedding space will result in a visualization located in the dashed square. Such an integrated view might be an exploded view in 3D space where the abstract hierarchical relationships are indicated through arrows.

2.5 Maximum Intensity Difference Accumulation

We now present another example of DVR-MIP integration in $\mathcal{A}$-space. This demonstrates that there are more than one possibilities to perform *interpolation* between points in $\mathcal{A}$-space. Maximum Intensity Difference Accumulation (MIDA) is a technique proposed by Bruckner & Gröller [3]. It is a volume rendering technique that integrates MIP and DVR. The integrated visualization preserves the complementary strengths of both techniques, i.e., efficient depth cuing from DVR and parameter less rendering from MIP. Since some datasets look better with DVR and others are best viewed with MIP, MIDA lets the user interpolate smoothly between DVR and MIP. Compared to the two-level volume rendering technique described in Section 2.1 MIDA is a spatially fine-grained integration approach and provides smooth transitions between the techniques. At each spatial position elements of both techniques are incorporated, whereas in two-level volume rendering algorithms are applied spatially disjoint. Figure 9(a) shows the result of using MIDA on an ultramicroscopy of a mouse embryo. Figure 9(b) shows the typical ray profiles generated with the different techniques.

2.6 Animated Transitions in Statistical Data Graphics

The Animated Transitions in Statistical Data Graphics proposed by Heer & Robertson [6] is a 2D to 2D integration performed in the information-visualization domain. The visualization techniques created provide smooth transitions between different visualizations of statistical data. The example reconstructed in $\mathcal{A}$-space



(a) (b)

Figure 9: (a) Ultramicroscopy of a mouse embryo showing the MIDA [3] rendering enlarged with possible *interpolations* from DVR to MIP in the bottom. (b) Shows typical ray profiles for (1) DVR, (2) MIDA and (3) MIP.

is an integration between scatter plots and bar charts. A benefit of this visualization is that the spatial relationship between sample points is visualized in the transition. The technique allows several different transitions between the statistical visualizations. Figure 10 shows in a schematic way two possible transitions from a scatter plot to a bar chart. This visualization technique is just one example of many 2D to 2D blending of algorithms approaches that exist.

2.7 Scattered Direct Volume Rendering

Our last somewhat speculative example from $\mathcal{A}$-space is the work by Rautek et al. [7] where a quite unusual integration is taking place. The integration is between 3D and 2D, specifically between DVR and scatter plots. On one side a volume rendering is shown and on the other side a scatter plot. In an animated transition the voxels are moving from the 3D volume-rendering space to their appropriate location in the 2D scatter plot and vice versa. Figure 11 shows two separate frames of the animated transition between DVR and scatter plots.

3 On the Nature of $\mathcal{A}$-Space

In the previous sections we have sketched the concept of $\mathcal{A}$-space. Via examples we have shown how visualization algorithms are points in $\mathcal{A}$-space and reconstruction is the process of blending between algorithms. This concept entails more than

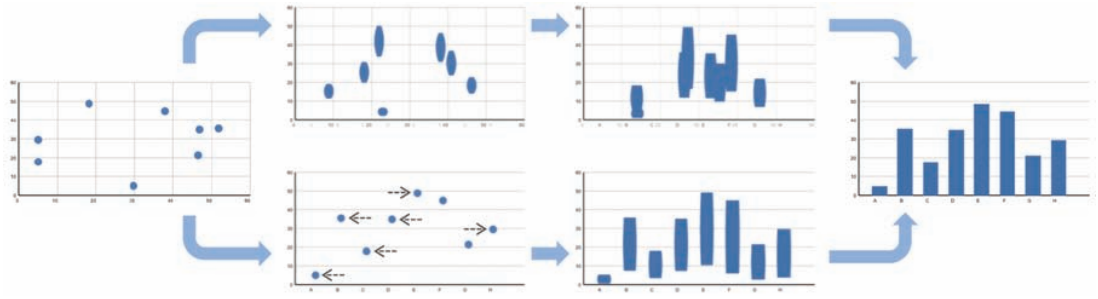


Figure 10: A schematic overview showing two possible transitions from a scatter plot to a bar chart [6].

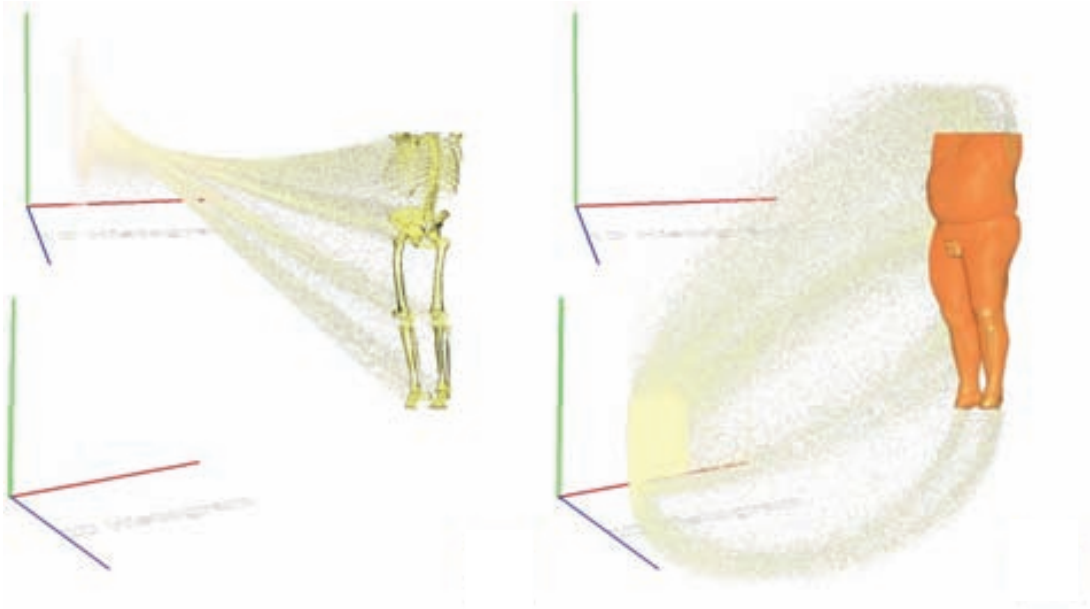


Figure 11: Two separate frames of the transition from DVR to scatter plots. [7]

that though and we will in this section indicate further aspects of $\mathcal{A}$-space and open issues. $\mathcal{A}$-space is not a space in the strict mathematical terms. It shall act as a thought-provoking concept. Real-world phenomena are increasingly measured through several heterogeneous modalities with quite different characteristics. We believe that this increased data complexity can be tackled through integrated views. $\mathcal{A}$-space may help to more systematically explore the possibilities to blend together diverse visualization approaches with complimentary strengths. In the following we shortly discuss various open issues concerning $\mathcal{A}$-space.

Interpolation and reconstruction In the examples we have shown various types of blending algorithms together. Sloppily we have called this blending interpolation and reconstruction. Can other types of interpolation be trans-

ferred to $\mathcal{A}$-space? Would it be possible to use barycentric coordinates to smoothly and simultaneously interpolate between three or more algorithms? Having several data sources for the same phenomenon makes fusion often a necessity. Fusion at the data and image level have been around for a long time. The visualization pipeline, however, consists of many more steps from the data to the final image. It is here where algorithm fusion and $\mathcal{A}$-space come into play. We cannot fight increased data complexity with increased visual complexity. A sensible step would be to move therefore from linked to integrated or combined views.

Dimensionality and units What is the dimensionality of $\mathcal{A}$? What are the dimensions of $\mathcal{A}$? Would knowing the *coordinates* of an algorithm give some insight into the possibilities of reconstruction or the compatibility of algorithms. For example do algorithms in the same plane share some features? What kind of units does $\mathcal{A}$-space use? Is $\mathcal{A}$ a metric space? What is the distance between two algorithms and how does one measure this distance? Currently algorithms are often categorized according to their spatial and temporal asymptotical complexity. Could visualization algorithms be categorized according to other measures like visual complexity, number of algorithms integrated, algorithm length? Does $\mathcal{A}$-space have a set of basis algorithms where all other visualization techniques can be reconstructed from?

Transformations Which transformations make sense in $\mathcal{A}$-space? Let us assume we are starting with a linked view as the initial visualization where all the component algorithms are known points in $\mathcal{A}$-space. Is it possible to create a generic transformation that would convert such a visualization into an integrated visualization based on the linking between the views? Would that process reconstruct a new point in $\mathcal{A}$-space or create a mapping to an already known point?

Iso-algorithms Iso-surfaces are very important in the scientific-visualization domain. Analogously are there iso-algorithms in $\mathcal{A}$-space, with the visual complexity as iso-value for example?

Sub-spaces Into what sub-spaces can $\mathcal{A}$-space be subdivided? Would the subspaces correspond to natural categorizations such as 2D and 3D techniques or information visualization and scientific visualization techniques?

Local neighborhood Given a point in $\mathcal{A}$-space how would the local neighborhood look like? Example measures might be gradient, divergence, curl. What would be the gradient, divergence or curl of a DVR or MIP, i.e., ∇DVR, ∇MIP?

Interaction Given the typically dense overlapping in integrated views, interaction has hardly been explored in this context. An interaction event might simultaneously navigate in several spaces. How can the user be supported to efficiently interact with integrated views?

The above list of aspects and open issues of $\mathcal{A}$-space is for sure not complete. Creating integrated visualizations is currently done in an ad hoc way. Sparse data is reasonably simple to integrate, but the difficulty of integration increases with the density of the data. Dense data may have many important features collocated both spatially and temporally and currently there is no general way of solving this. Maybe some sort of exploded views in space and time could be an answer for this?

Categorizing the algorithms in $\mathcal{A}$-space may help in defining the boundaries of $\mathcal{A}$. The categorization may be to differentiate between fine and coarse visualization integration or to differentiate at what stage the integration is performed, i.e., data stage, algorithm stage or image stage.

4 Conclusion

Integrated visualization will become more important in the future. Integrated views are a not yet fully explored area and they are one answer to cope with increased data complexity. We have shown some of the possibilities in $\mathcal{A}$-space and we think it may be a new direction on how to look at ways to perform visualization integration. $\mathcal{A}$-space might be a useful tool for classifying and indicating the possibilities of integrated visualizations. There already exist many integrated visualizations that may benefit to be localized in $\mathcal{A}$-space. Increasing the population of $\mathcal{A}$-space would also indicate untapped regions where reconstruction is a possibility and could lead to new integrated visualizations. Fill in the holes of $\mathcal{A}$!!!

References

[1] J.-P. Balabanian, I. Viola, and E. Gröller. Interactive illustrative visualization of hierarchical volume data. To be submitted, 2009.

[2] S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller. Illustrative context-preserving exploration of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1559–1569, 11 2006.

[3] S. Bruckner and M. E. Gröller. Instant volume visualization using maximum intensity difference accumulation. *Computer Graphics Forum (Proceedings of EuroVis 2009)*, 28(3):775 – 782, 2009.

[4] M. Hadwiger, C. Berger, and H. Hauser. High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 40, Washington, DC, USA, 2003. IEEE Computer Society.

[5] H. Hauser, L. Mroz, G. I. Bischi, and M. E. Gröller. Two-level volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):242–252, 2001.

[6] J. Heer and G. Robertson. Animated transitions in statistical data graphics. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1240–1247, Nov.-Dec. 2007.

[7] P. Rautek and E. Gröller. Scattered direct volume rendering. Personal communication, 2009.

[8] M. Straka, M. Červeňanský, A. L. Cruz, A. Köchl, M. Šrámek, E. Gröller, and D. Fleischmann. The VesselGlyph: Focus & context visualization in CT-angiography. *IEEE Transactions on Visualization and Computer Graphics*, pages 385–392, Oct 2004.