Visualisierungsinstitut der Universität Stuttgart
University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3338

# Visualization of Space-Time-Structure of Electromagnetic Fields

Oliver Schmidtmer

| | |
|---|---|
| **Course of Study:** | Software Engineering |
| **Examiner:** | Prof. Dr. Thomas Ertl |
| **Supervisor:** | Dr. Sc. Filip Sadlo |
| **Commenced:** | May 14, 2012 |
| **Completed:** | November 13, 2012 |
| **CR-Classification:** | I.3.8, J.2 |

## Abstract

In the course of this work are techniques for visualization of time-dependant electromagnetic fields in space-time representation introduced. This happens on the basis of wave propagation, as this phenomenon is still not adequately researched. Therefore in this work are usually scalar fields used. The methods which are developed in this work could also be used for visualization of other typical wave propagation phenomena. In the case of vector field data, such as electromagnetic fields, they must be used on their magnitude. This methods also set a foundation for further developments in combination with analysis based on vector topology of electromagnetic fields, for further more detailed visualizations.

The in the course of this work introduced methods base on the extraction of height ridge and valley surfaces. To avoid occlusions in space-time representation and further reduce the complexity of the representation, so-called virtual sources are introduced in the visualisation of wave phenomena. For this are two approaches on extracting these virtual sources explored and different approaches on processing them to space-time curves. By using these space-time-curves for a reconstruction of the original field, they deliver a compact representation of the field. This allows an effective visualisation and interpretation of the space-time structures in the field and their relations.



## Kurzfassung

Im Rahmen dieser Arbeit werden Techniken zur Visualisierung zeitabhängiger, zweidimensionaler elektromagnetischer Felder in einer Raum-Zeit-Darstellung vorgestellt. Dies geschieht auf Basis der Wellenausbreitung, da dieses Phänomen noch nicht angemessen erforscht wurde. In dieser Arbeit wird daher vorwiegend mit Skalarfeldern gearbeitet. Die hier entwickelten Methoden können allgemein zur Visualisierung der Wellenausbreitung verwendet werden und müssen im Fall von Vektorfeldern auf deren Magnitude angewendet werden, wie dies bei elektromagnetischen Feldern der Fall ist. Die entwickelten Methoden bieten auch eine Basis für Weiterentwicklungen im Zusammenspiel mit Analysen auf der Vektortopologie der elektromagnetischen Felder, für weitere, detailliertere Visualisierungen.

Die in dieser Arbeit vorgestellten Methoden basieren auf der Extraktion von Grad- und Talflächen. Um Verdeckungen in der Raum-Zeit-Darstellung zu vermeiden und die Komplexität der Darstellung weiter zu reduzieren, werden so genannte virtuelle Quellen in die Visualisierung der Wellenausbreitung eingeführt. Dabei werden zwei Ansätze zur Extraktion dieser virtuellen Quellen vorgestellt und verschiedene Ansätze um diese zu Raum-Zeit-Kurven zu verarbeiten. In dem diese Raum-Zeit-Kurven als Basis einer Rekonstruktion des ursprünglichen Feldes verwendet werden, liefern sie eine kompakte Repräsentation des Feldes, was eine effektive Visualisierung und Interpretation der räumlich-zeitlichen Strukturen und Zusammenhänge erlaubt.

# Contents

# List of Figures



# List of Algorithms

# 1. Introduction

## 1.1. Motivation

The domain of time dependant electromagnetic field analysis and visualization is a huge and still not adequately covered field. Most approaches rely on comparably simple visualization techniques like isosurface extraction or volume rendering. More complex approaches investigate on topological structures of these fields. Many researches are done and heading in the direction of vector field analysis on using specialized topology analysis for peculiarities on electromagnetic fields.

In the course of this work another direction is pursued. It focuses on the extraction of features in the space-time representation of time-dependant electromagnetic fields. Thereby the focus is laid on wave propagation, because no appropriate visualization approaches on this exist so far and wave propagation is the most significant phenomenon in time-dependant electromagnetic phenomena. The approaches in this work provide a simplified representation of electromagnetic fields phenomena, including reflections and superpositions effects.

Our approach starts with the height ridge surface extraction from the space-time representation of two-dimensional fields. Note that it is usable on any scalar field of wave phenomena. In the case of electric or magnetic fields, or even other vector fields, the methods have to be applied to their magnitude. Although the extracted ridge surfaces in space-time already reduce the amount of data for visualization by far and provide a valuable visualization of time-dependant electromagnetic field phenomena, they still suffer from occlusion and clutter. Therefore further approaches on the extraction of so-called virtual sources are done, which are represented by curves in space-time. Those virtual sources can represent true sources of the original data, but can also originate in effects such as superpositions, reflections or other disturbances of the field. Finally we reconstruct the signal of those virtual sources, in terms of which signal strength along the space-time curves reproduces the original field as close as possible. The resulting visualization by colored space-time curves of virtual sources together with context information provided by selected ridge surfaces or volume rendering provides a direct notion of the causes that are responsible for the observed field at a given location. It is expected, that virtual sources which represent true sources of the original data should provide a clearer and more ridge aligned signal, than sources which originate in other effects.

## 1.2. Related Work

A major part of analysis and visualization of electromagnetic fields is conducted in the domain of vector field topology. Sanderson et al. [SCT+] research for recurrent patterns in toroidal magnetic fields. Their application for research is the magnetic structure of plasma in tokamak fusion reactors. Bachthaler et al. [BSW+12] extend traditional vector field topology visualization to magnetic flux in two-dimensional magnetic fields. This provides a quantitative view on the magnetic flux and helps thereby on identifying magnetic rings and magnetic chains. Machado et al. [GMME12] research visualization of the electromagnetic fields in terms of the sun. Their targets are magnetic effects from features of the corona, such as coronal loops.

Research on wave propagation visualization is more common on acoustic data than on electromagnetic fields. Yokota et al. [YST] develop a visualization for sound propagation and scattering in rooms based on wave boundaries in a two-dimensional sound field. A similar approach is done for evaluating acoustical environments in enclosed three-dimensional space for reflection analysis by Omoto and Uchida [OU]. Obermaier et al. [OMD+] use mesh-free valley surface extraction on low frequency sound simulations. This is the only work that we are aware of, which uses ridge and valley extraction in the context of wave propagation visualization. Note that they use they use it in a three-dimensional space field, while we use a space-time representation of two-dimensional fields and extract features therefrom. Deines et al. use phonon tracing for simulation and visualization of sound waves in [BDM+], [DBM+06] and [Dei08]. An other approach for interactive purposes of sound propagation is using frustum tracing in [LCM07] by Lauterbach et al. For propagation time and sound clarity analysis and visualization a ray based approach is used by Stettner and Greenberg in [SG89]. Simulation and visualization of sound strength for acoustic design is researched by Monks et al. in [MOD00].

Height ridges, as used for the research this work, are defined by Eberly in [Ebe96]. Eberly defines ridges on the basis of the gradient $g$, eigenvalue $\lambda$, and eigenvectors $\vec{e}$ of the Hessian matrix $\mathbf{H}$. A ridge extraction method, called Marching Ridges, is researched in [FP98] and [FP01] by Furst and Pizer. Their method however does not follow a marching cubes approach, but employs contour tracing from a user-defined starting point. Sadlo and Peikert advance on this with a marching cubes approach for ridge surface extraction that uses adaptive mesh refinement for efficient extraction in [SP07].

An other research by Peikert and Sadlo presents a more efficient approach which does not need explicit eigenvector calculation in [PS08]. Their approach uses the parallel vectors operator [PR99] for extracting ridges. In their approach Ridges are extracted using a matrix determinant constructed from the gradient and the Hessian. Further studies on crease surface extraction which expand on [PS08] are done by Schultz et al. in [STS10]. Lindeberg also provides techniques on edge and ridge detection in [Lin96] on two-dimensional imaging. Damon researches a solution for connecting disjointed ridge lines which result from transitions around singular Hessian points in [Dam]. In the field of diffusion tensor magnetic resonance imaging, ridge lines and surfaces are used for analysis by Kindlmann et al. [KTW06]. Furst et al. [FKMP] show that features of images, such as boundaries and skeletons, can be formulated as height ridges in an extended Euclidean space. Research on surface extraction of vortex and strain

skeletons is done by Sahner et al. in [SWTH07]. A research on feature flow fields using ridge lines in three-dimensional data is done by Theisel and Seidel in [TS03].

Other approaches include the approach on particle-based ridge and valley extraction by Kindlmann et al. in [KSSW09] and the research on finding ridges and valleys on discrete surfaces by Soo-Kyun Kim and Chang-Hun Kim in [KK05] and on dense triangle meshes by Ohtake et al. in [OBS04]. However, those approaches do not directly relate to height ridges.

The concept of space-time visualization is already a successfully used concept. It was used by by Weinkauf et al. [WSTH07] for vortex extraction from fluid flows, by Machado et al. [GMME12] for visualizing solar dynamics data and by Bachthaler et al. [BSDW12] for extracting topological structures in time-dependant 2D vector fields.

## 1.3. Structure

The process on this work was not strictly linear. For example, the process of source extraction required refining back in the ridge extraction stage and even new filtering approaches for the ridge surface to suppress the creation unnecessary virtual sources. The same also happened for the signal reconstruction stage with respect to the virtual source extraction stage. In the whole however, each new step introduced new perspectives for the further processing in the subsequent steps. Therefore the transcription of this work is structured in following way:

**Chapter 2 – Basics:** This chapter contains a description of the requirements and limitations of this work and introduces required fundamentals.

**Chapter 3 – Dataset Generation:** The topic of this chapter is the generation of the datasets, which are used for testing and validating the developed methods and also for illustration purposes.

**Chapter 4 – Wavefront Extraction:** Presents basic feature extraction approaches based on wavefront surfaces in space-time.

**Chapter 5 – Extraction of Virtual Sources:** Describes how virtual sources are extracted from previously extracted wavefront surfaces.

**Chapter 6 – Virtual Source Signal Reconstruction:** Discusses approaches on the reconstruction of signals of the virtual sources from the original dataset, which could provide a compact space-time representation.

**Chapter 7 – Results:** Presents results on using the approaches which are introduced from this work.

**Chapter 8 – Conclusion:** In this chapter the results of this work are discussed and examples for possible further work are given.

**Appendix A – Implementation:** Provides a documentation of the the implementation details of developed tools and filters.

# 2. Basics

In the process of this work a few preconditions have to be met for a successful visualization of the addressed data. In particular, requirements to dataset resolution and data sizes have to be met to assure successful analysis. It includes how electromagnetic fields are handled with the presented techniques, as they are vector fields and the major part of this work addresses wave propagation in scalar fields. Here is also referred to the external libraries and tools, which were used to accomplish this work.

## 2.1. Handling of Electromagnetic Fields

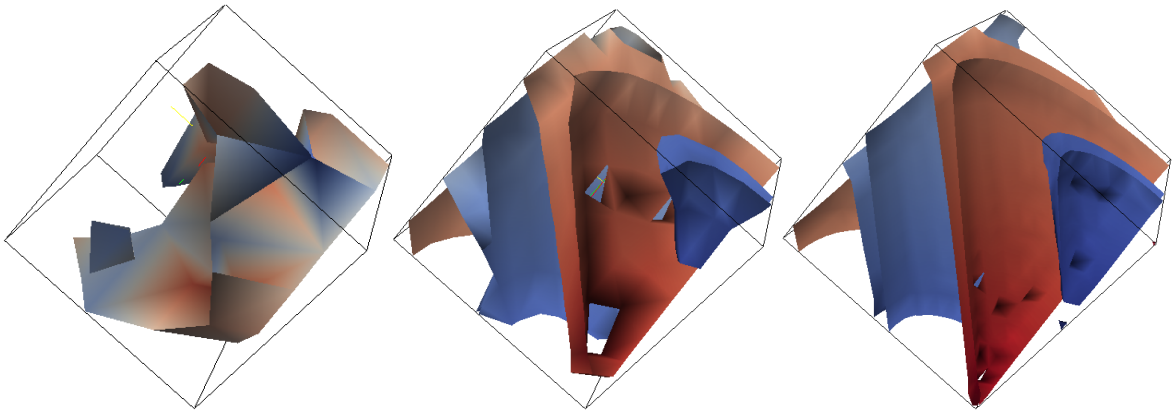
Electromagnetic fields consist of two three-dimensional vectors at each point in space. These represent their two components, the magnetic field $B$ and the electric field $E$. Hence those vector fields have to be converted to scalar fields first. This is simply done by calculating the magnitude of both component vectors separately and by using both resulting scalar fields for separate analysis.

## 2.2. Space-Time Representation

The target of this work is the feature extraction from time-dependant two-dimensional fields. This representation is obtained by stacking the two-dimensional data for providing a time axis $t$ that is perpendicular to the spatial $x$- and $y$-axes. As a result the obtained three-dimensional datasets contain an original two-dimensional scalar field at each time-coordinate and the resulting representation is rendered by treating the time axis as $z$-axis. In the context of dataset generation (Chapter 3), from three-dimensional field sources, such as the Hertzian dipole, only a slice of the time-dependent field is used and stacked in space-time.

## 2.3. Sampling Requirements

To enable successful feature extraction from a given dataset, the dataset has to meet certain sampling rates. They depend on the maximum frequency $f$ of oscillations in the field and the phase velocity $v_p$ of the given wave type in the traversed medium. From the wavelength $\lambda = v_p/f$ the sampling rate for the spatial dimensions can easily be obtained. The absolute maximum for the sampling distance is $\lambda/8$, better $\lambda/16$ for accurate ridge surface extraction. In a similar fashion this is calculated for the temporal resolution as $\Delta t = 1/f$ and the requirement



(a) (b) (c)

**Figure 2.1.:** Comparison of sampling rates in datasets. The same source is sampled with sampling rates of $\lambda/4$ (a), $\lambda/8$ (b), and $\lambda/16$ (c). The surfaces are extracted with the method as described in Section 4.3.

for the sampling distance with $\Delta t/8$, respectively $\Delta t/16$. Figure 2.1 shows a comparison between different sampling rates. It can be seen, that sampling distances higher than $\lambda/8$ do not allow a clear wavefront surface extraction while $\lambda/8$ still exhibits holes even in simple datasets. Therefore for the subsequent processing steps after surface extraction higher sampling rates are necessary.

## 2.4. Requirements regarding Dataset Extents and Sizes

With the already mentioned required sampling rates the size of the analyzed region also has to be limited in relation to wavelength and frequency. In the example of a 2.4GHZ electromagnetic wave, with sampling rates of $\lambda/16$, $\Delta t/16$, a time interval of a second would require $\sim 384 \cdot 10^8$ samples. In the spatial dimensions this would require $\sim$128 samples per meter.

In regard to the subsequent techniques for signal reconstruction, datasets should cover around double the time duration of what a wave would need to traverse the spatial extent of the dataset. This ensures that random sample points in space-time have good chances that their observed signal originates from sources within the dataset time range. Otherwise, in wide datasets that are flat in the time extent, most sampled points would refer to sources long time before the dataset begins.

## 2.5. Eigen - Linear Algebra Library

For the signal reconstruction, which is presented in Chapter 6, the least squares approach is used. An introduction to least squares and solving methods is, for example, provided in

[Bjö]. These calculations can easily require the calculation of matrices with many hundreds or thousands of columns. Despite most coefficients in the matrices being zero, standard dense matrix implementations would require too much memory. The use of intelligent sparse matrices, that only store nonzero values, is necessary, together with appropriate decomposition methods. Therefore the library Eigen[1] is used.

## 2.6. VTK - Visualization Framework

For the data management and processing the Visualization Toolkit (VTK)[2] is used. With this it is possible to use already existing and well validated modules for data loading, saving, and a multitude of filtering techniques. Therefore the techniques presented in this work are implemented as custom filter modules in VTK.

The resulting representations could then be, for example, visualized with ParaView[3]. ParaView provides a graphical user interface for all aspects of VTK and can be used for flexible visualization configuration and generation of animations from the analyzed datasets. This is faster in configuration and everyday use and therefore more flexible than directly coding the pipeline with VTK.

The custom filtering modules which are developed in course of this work however are not implemented for usage within ParaView directly. This implementation of the modules as ParaView plugins was avoided to ease development and to save time. The conversion from VTK filters to ParaView plugins however would be straightforward with sufficient knowledge of ParaView and can be carried out as future work. Therefore the custom modules can so far only be used in custom VTK programs and their output can be loaded into ParaView.

[1] http://eigen.tuxfamily.org/
[2] http://www.vtk.org/
[3] http://www.paraview.org/

# 3. Dataset Generation

A first task for this work was the creation of datasets, on which the feature extractions could be developed, validated, and illustrated. As a broad variety of datasets of different complexity was required, a standalone application including a GUI (Figure 3.1) was developed. With the help of this application sources can be placed in two-dimensional space. Second and an as important part of the application is the selection of space and time ranges and sampling rates in a region of interest, for which the data should be generated. The result can be exported as VTK Image data (.vti) in the form of a structured grid. The application was originally written for sampling simple wave equations and later on extended with Hertzian dipoles. In the following sections both analytical solutions for the fields are described, together with their parameters.

Additionally, for an evaluation on more realistic datasets, datasets were obtained using a finite-difference time-domain method by Siphos and Thompson [ST08], using an existing implementation by Thomas Müller.

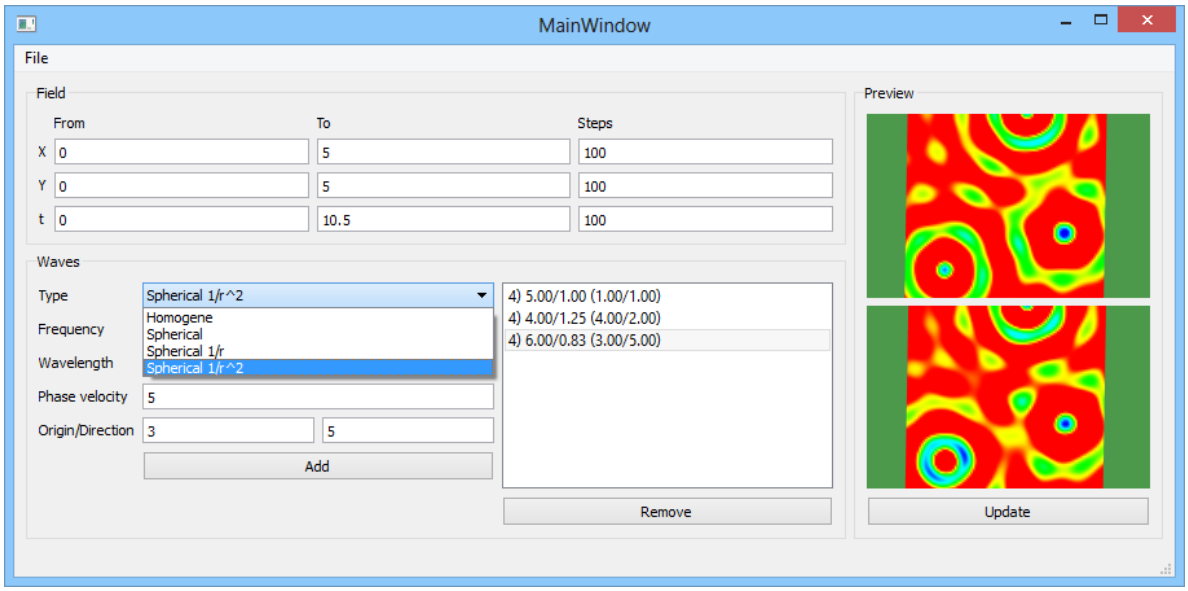



**Figure 3.1.:** Dataset generation for scalar fields. Simple preview at $t_{From}$ and $t_{To}$ time slices on the right.

## 3.1. Wave Equation

For experimental purposes the application supports different wave types. Basic inputs for the formulation of all types are the frequency $f$, wavelength $\lambda$, and time $t$. Further variables are the vectors $\mathbf{x}$, which defines the current position in space, $\mathbf{o}$ the origin of the source, and $\mathbf{d}$ a unit vector of the direction of the wave. From those are the angular frequency $\omega = f \cdot 2 \cdot \pi$ and the wavenumber $k = 2 \cdot \pi / \lambda$ calculated.

**Linear wave, no falloff** A wave from a directional source, no range falloff.

$$\cos(\langle \mathbf{d}, \mathbf{x} \rangle \cdot k - \omega \cdot t)$$

**Point source, no falloff** A wave originating at a point source, no range falloff.

$$\cos(\|\mathbf{x} - \mathbf{o}\| \cdot k - \omega \cdot t)$$

**Point source, quadratic falloff** A wave originating at a point source, quadratic range falloff.

$$\frac{\cos(\|\mathbf{x} - \mathbf{o}\| \cdot k - \omega \cdot t)}{\|\mathbf{x} - \mathbf{o}\|^2}$$

## 3.2. Hertzian Dipole

For the generation of electromagnetic fields a Hertzian dipole as in [Kar, p. 224] is used. As the analytical solution uses spherical coordinates, the source origin $\mathbf{o}$ and sample position $\mathbf{x}$ have to be converted by equation 3.1.

$$\begin{pmatrix} r \\ \Theta \\ \varphi \end{pmatrix} = \begin{pmatrix} \|\mathbf{x} - \mathbf{o}\| \\ \operatorname{acos}(\mathbf{x}_z - \mathbf{o}_z) / \|\mathbf{x} - \mathbf{o}\| \\ \operatorname{atan2}(\mathbf{x}_y - \mathbf{o}_y, \mathbf{x}_x - \mathbf{o}_x) \end{pmatrix} \tag{3.1}$$

In addition to the previously defined parameters are the dipole moment $p$ and the zero phase $\alpha$ used. Further required are the constants for permittivity $e = 8.85418781762 \cdot 10^{-12} F/m$ and speed of light $c = 299792458 m/s$. To simplify the formula, the substitution $\tau = \omega \cdot (t - r/c) + \alpha$ is used.

The resulting fields $\vec{E}$ (electric) and $\vec{H}$ (magnetic) of Equations 3.2 and 3.3 are also given in spherical coordinates.

$$\vec{E} = \begin{pmatrix} \frac{2 \cdot p \cdot cos(\varphi)}{4 \cdot \pi \cdot e} \cdot (\frac{1}{r^3} \cdot \sin(\tau) + \frac{\omega}{c \cdot r^2} \cdot \cos(\tau)) \\ 0 \\ \frac{p \cdot sin(\varphi)}{4 \cdot \pi \cdot e} \cdot ((\frac{1}{r^3} - \frac{\omega^2}{r \cdot c^2}) \cdot \sin(\tau) + \frac{\omega}{c \cdot r^2} \cdot \cos(\tau)) \end{pmatrix} \tag{3.2}$$

$$\vec{H} = \begin{pmatrix} 0 \\ \frac{\omega \cdot p \cdot \sin(\varphi)}{4 \cdot \pi} \cdot (-\frac{\omega}{c \cdot r} \cdot \sin(\tau) + \frac{1}{r^2} \cdot \cos(\tau)) \\ 0 \end{pmatrix} \tag{3.3}$$

Those are converted back to Cartesian coordinates with the Equations 3.4 and 3.5.

$$\vec{E}_{Cartesian} = \begin{pmatrix} \vec{E}_r \cdot \sin(\varphi) \cdot \cos(\Theta) + \vec{E}_\varphi \cdot \cos(\varphi) \cdot \cos(\Theta) \\ \vec{E}_r \cdot \sin(\varphi) \cdot \sin(\Theta) + \vec{E}_\varphi \cdot \cos(\varphi) \cdot \sin(\Theta) \\ \vec{E}_r \cdot \cos(\varphi) - \vec{E}_\varphi \cdot \sin(\varphi) \end{pmatrix} \tag{3.4}$$

$$\vec{H}_{Cartesian} = \begin{pmatrix} -\vec{H}_\Theta \cdot \sin(\Theta) \\ \vec{H}_\Theta \cdot \cos(\Theta) \\ 0 \end{pmatrix} \tag{3.5}$$

# 4. Wavefront Extraction

As the visualization of the whole field, e.g., by means of volume rendering, would suffer from massive occlusion, it is our goal to only extract the wavefronts in form of generalized extrema, called ridges. Hence our first step consists in the extraction of those ridge surfaces. Figure 4.1 illustrates the ridge concept for a one-dimensional function, distinguished to ridges and valleys. In the instance of a single point source for the waves, the result of a codimension-one ridge extraction would be a circle in a two dimensional field or a sphere in a three-dimensional field.

As this work addresses two-dimensional fields with stacked time dimension, i.e. in three-dimensional space-time representations, the expected result for the previously assumed point sources would be a funnel-like structure for each extracted wavefront. In the following different approaches for visualization and extraction are explained.

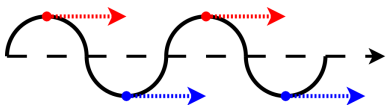
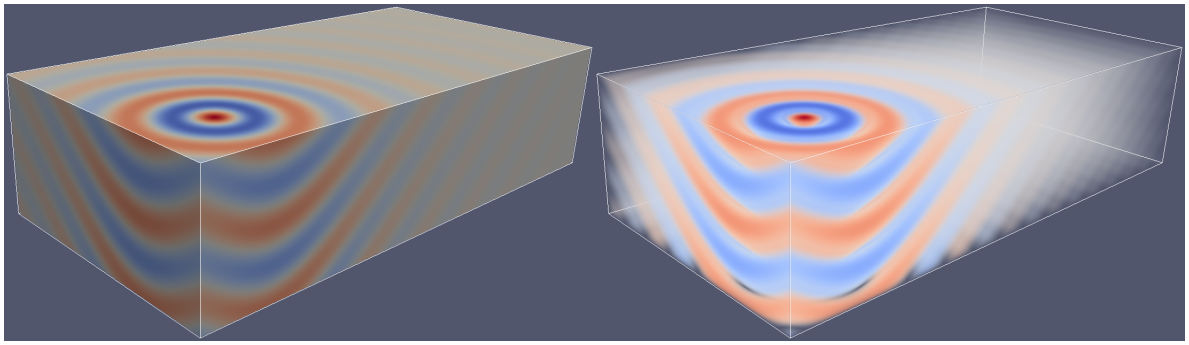


**Figure 4.1.:** Illustration of ridges and valleys for a sinus wave travelling to the right. Ridge points are marked red, valleys blue.

## 4.1. Basic Visualization

As a first approach for the wavefront visualization, a simple boundary surface display and volume rendering, as shown in Figure 4.2, are used. With the structures visible on the outside, we complemented this visualization by volume rendering to see inside. Mapping the absolute scalar values to opacity, the wavefronts appeared as distinguishable dense structures with empty spaces between. However, the volume rendering approach suffers strongly from perception issues and visual clutter, representing a method for quick overview but insufficient for detailed investigation.

With increasing distance to the source, the wavefront structures are thinning out to indistinguishable fog, as the ridges and valleys absolute values become lower. It became apparent that instead of the scalar values a feature extraction approach could compensate for the flattening of the waves on their way from the sources. Furthermore, the results from volume rendering could not be used for further analysis, as it delivers no geometric representation, only an image. Therefore, volume rendering can only serve for a first overview of a dataset or to provide context to results from the feature extraction techniques, which are explored in this work.



**(a)** **(b)**

**Figure 4.2.:** Color coding on domain boundaries and volume rendering.

## 4.2. Ridges

Common definitions and works on ridges base on that by Eberly [Ebe96]. Eberly uses the gradient $g$ of the scalar field and the eigenvalues $\lambda$ as well as the corresponding eigenvectors $\vec{e}$ of the Hessian matrix of the scalar field. On a $n$-dimensional field with the Hessian eigenvalues $\lambda_1 \geq \lambda_{...} \geq \lambda_n$ ridges are defined by, according to this definition, points where $\lambda_n < 0, \langle \vec{e_n}, g \rangle = 0$, valleys accordingly where $\lambda_1 > 0, \langle \vec{e_1}, g \rangle = 0$.

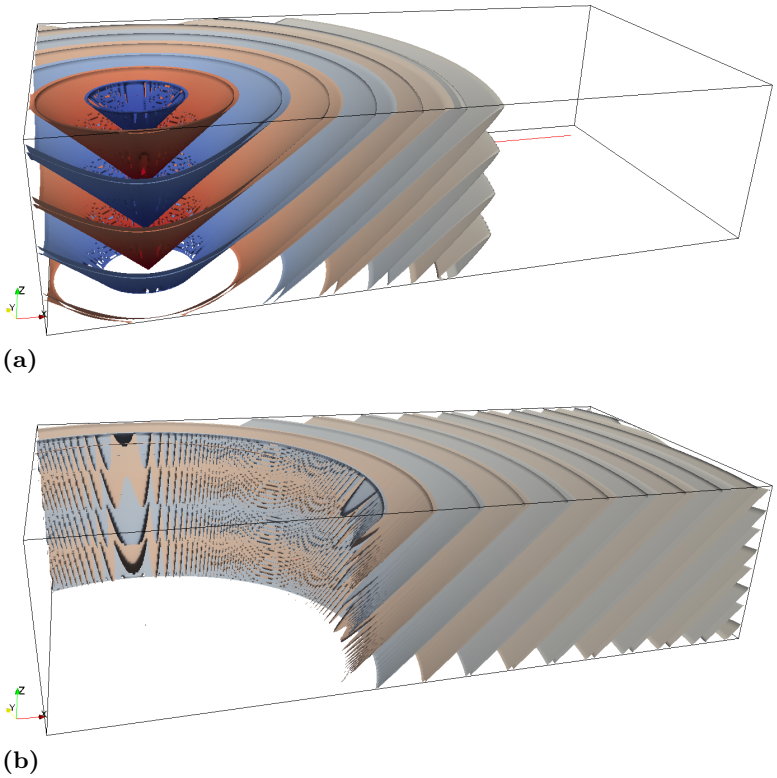
## 4.3. Gradient-Magnitude-Based Ridge Extraction

The idea behind this approach is based on methods from analyzing one-parametric functions. For a function $f(x)$ the first derivative $f'(x)$ has to vanish if $x$ is part of a crease. Hence, $f'(x) = 0$ solves for all extrema, including minima, maxima and saddles.

As a generalization for three-dimensional scalar fields, the magnitude of the gradient can be used. Hence, crease surfaces can be formulated inside a scalar field $S$ according to Equation 4.1.

$$\|\nabla S\| = 0 \tag{4.1}$$

In a discretized implementation this implies the calculation of the gradient from central differences and calculating the magnitude. The extraction of this set of points represents an isosurface extraction problem, which can be achieved by means of the marching cubes algorithm [LC87] with a isovalue of zero. In generic discretized scalar fields, however the isosurface at level zero is hard to extract because there are no negative gradient magnitudes. Therefore a tolerance to zero $\varepsilon$ has to be introduced. As the necessary $\varepsilon$ relates to the values at the ridges, it must be determined for each scalar field independently.



**(a)**

**(b)**

**Figure 4.3.:** The source dataset using a sampling distance of $\lambda/50$ and $\varepsilon = 0.6$ in (a) and $\varepsilon = 0.1$ in (b) for extraction.

In Figure 4.3 it can be seen, that $\varepsilon$ must even be reconsidered for focusing on different parts of the field. While in (a) the inner most ridge parts exhibit gaps within the surface, because the $\varepsilon$ is too low with respect to the gradient magnitude on the ridges, the outer ridge parts are insufficiently represented by two distant surfaces. With the lower $\varepsilon$ in(b) the flattening outer ridge regions are captured, but at the cost of disruption of the inner-most ridges. It generally applies, that the higher the sampling rate is, the lower the $\varepsilon$ can be and the better the ridges are captured. However, this would require very high sampling rates for practical usage in datasets.

An other disadvantage of this method is, that it cannot directly distinguish which surfaces are ridges and which are valleys. As superposition of waves could raise or lower ridge values, the sign of the original scalar could not be used for distinction.

## 4.4. Marching Ridges

The second approach for ridge surface extraction uses an algorithm proposed in [SP07] and [FP01]. The idea behind this algorithm is to use a modified marching cubes algorithm based

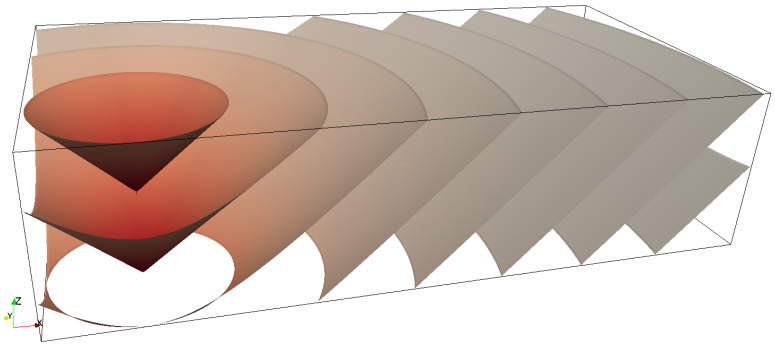


**Figure 4.4.:** Same Dataset as in fig. 4.3, using Marching Ridges with ridges only.

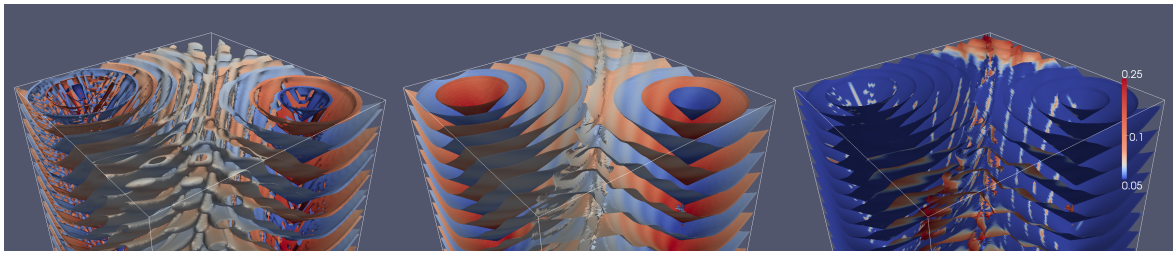
on the Eberly criterion (Section 4.2) in terms of the first and second derivatives of the scalar field.

As mentioned in Section 4.3, zero isosurfaces could be detected using $\|\nabla S\| = 0$, but this approach would not distinguish them into ridges and valleys and would suffer from sampling problems. Therefore the second derivative, the Hessian matrix, is used. The eigenvalues $\lambda$ of the Hessian contain the second derivatives along the corresponding eigenvectors. So for a ridge the minor eigenvalue $\lambda_{min}$ is determined and for a valley the major eigenvalue $\lambda_{max}$. In addition to $\|\nabla S\| = 0$ the criterion for ridges then is $\lambda_{min} < 0$ and $\lambda_{max} > 0$ for valleys. This, however, would exhibit the same problems with sampling and finding points where $\|\nabla S\| = 0$ as the method described in Section 4.3.

Therefore the central part of the algorithm is the approach used to obtain a criterion to decide at which point between the samples the gradient and the Hessian should be interpolated. Here the eigenvector $\vec{ev}$ of the previously determined eigenvalue $\lambda$ is used, which points in the direction of strongest curvature. The dot product from gradient and eigenvector in this point, $\langle \vec{ev}, \nabla S \rangle$, could be used, as the orientation of the curvature should stay constant around a ridge, but the gradient should change direction, what produces a negative value on one side of the ridge and a positive on the other side. So the derivative in eigenvector direction $\langle \vec{ev}, \nabla S \rangle$ is used as scalar field in the respective marching cubes cell together with an isovalue of zero to determine the ridge.

However, as eigenvectors define a direction without orientation and as a result two possible directions, they must be consistently oriented for each cell before determining the marching cubes case. For this purpose a covariance matrix is calculated, which uses both directions of the eigenvectors belonging to the cell. Then for each point the dot product between the respective eigenvector $\vec{ev}$ and the PCA-eigenvector of the major eigenvalue of the covariance matrix is used. If this dot product is negative, the respective eigenvector is flipped.

The pseudocode in Algorithm 4.1 outlines the algorithm, but lacks details regarding the used filtering methods. As previously mentioned, the first necessary filtering in the marching cubes cases step is to discard all triangles, where at least at one vertex the minor eigenvalue $\lambda_{min}$ is not below zero in the case of ridge extraction or the major eigenvalue $\lambda_{max}$ not over zero if valleys are extracted. Based on this filter, an offset around zero could be used for




(a) (b) (c)

**Figure 4.5.:** Comparison from left to right: Gradient-magnitude approach, marching ridges and Euclidean distance. The color scale in (c) refers to the cell width of 0.5 and indicates differences between one to five cell widths.

suppressing flat ridge or valley regions, because stronger ridges have higher absolute eigenvalues $\lambda$. In the post processing step, the connected component sizes of the resulting ridge surface representation are a good filter. In Figure 4.4 all components with less than 1000 connected triangles were rejected. For fragments due to superpositions, fringe cutting has shown effective. On fringe cutting are all triangles cropped, which have at least one edge that is not connected to an other triangle. This filter could be used for multiple iterations for further smoothing.

A range filtering for ridges and valleys by the scalar field value can also be used and has a similar effect like the eigenvalue threshold. However, it must be set for ridges and valleys independently and is dependent of the ground level around which ridges and valleys oscillate. Further this can vary in a dataset due to superpositions.

## 4.5. Comparison

In comparison, the marching ridges algorithm provides a much better precision than the gradient magnitude method. As this is necessary for further analysis, it is used in this work for further steps. It however has a much higher computing time. Therefore, if only a fast overview is needed, the gradient magnitude method has its benefits.

On a more complex dataset with two sources, as in Figure 4.5, further differences can be seen. Where marching ridges shows flattened, deformed ridges and jags on superpositions (b), the gradient-magnitude approach splits to tube like structures (a). This can be explained as separations from the ridge surface through the superposition of the other source. The tube then emerges from the $\varepsilon$ tolerance to zero of the gradient-magnitude approach. With further distance to the interfering source they would connect back to the main ridge. As it can be seen on the Euclidean distance graphic (c), both produce similar results and diverge on the superpositions in the middle of both sources.

**Algorithm 4.1** Ridge / Valley Extraction Algorithm

```
procedure ExtractSurface(ScalarField S, type ∈ {ridge, valley})
    EigenvectorField E
    GradientField G
    HessianField H
    calculateGradient(in S, out G)
    calculateHessian(in G, out H)
    for all Point ∈ S do
        Eigenvalue eigenVal
        if type == ridge then
            getLowestEigenvalue(in H[Point], out eigenVal)
        else if type == valley then
            getHighestEigenvalue(in H[Point], out eigenVal)
        end if
        calculateEigenvector(in eigenVal, out E[Point])
    end for
    for all Cell ∈ S do
        Matrix C
        Eigenvector eigenVec
        calculateCovarianceMatrix(in E[Point ∈ Cell], out C) // Using E and −E of
all Points ∈ Cell
        if type == ridge then
            getLowestEigenvalue(in C, out eigenVal)
        else if type == valley then
            getHighestEigenvalue(in C, out eigenVal)
        end if
        calculateEigenvector(in eigenVal, out eigenVec)
        for all Point ∈ Cell do
            if ⟨E[Point], eigenVec⟩ > 0 then
                E[Point] = −E[Point]                    // flip vector orientation
            end if
        end for
        calculateMarchingCubeCase(in ⟨E[Point], G[Point]⟩ ∈ Points ∈ Cell, iso-
value = 0)
    end for
end procedure
```

# 5. Extraction of Virtual Sources

After the extraction of ridge and valley surfaces, those informations can be used to locate virtual sources of waves. Obviously, strong real sources are identifiable from a funnel like structure (Chapter 4). A drain would look like the reverse. For sources which are weaker as other sources around and whose ridges are deformed and displaced by superpositions, this is not as easy.

In visualization it is impossible to decide from the data, whether a indicated source is a real source or origins from superpositions. Therefore in this work the extracted sources are called virtual sources and used as an indicator of the features of the original scalar field. In the following are two possible methods for virtual source extraction investigated.

## 5.1. Extraction by Local Minima of Ridge Surfaces

As mentioned, ideal sources form funnels with the source position at the bottom. So it is obvious to search for minima at the time axis on the ridge surfaces. Because ridges of multiple sources can connect to a single surface, the search for minima must be local and not global.

As deformations and inaccuracies through sampling can cause false positives it is worthwhile to use at least a two ring neighbourhood as a minimum condition, i.e., we require that all neighboring vertices of a minimum vertex candidate reside at later times, in terms of a location above the current candidate on the time axis. Our experiments have shown that a greater neighbourhood search does not substantially reduce the further count of false positives in relation to computing cost. However, through modifying the ridge surfaces, using fringe cutting as a filter, the count of false positives around deformations from superpositions could be significantly reduced. Further, a condition on the surface for discarding the found minima can be used. If a local minima vertex has triangles, whose edges are not all connected to other triangles, i.e., it is located at the boundary of a ridge surfaces, it is probable that the minimum is a result of deformations, superpositions, or boundary effects. Therefore it is usually discarded in our approach.

As good datasets contain multiple ridge and valley wavefronts outgoing from each source, missing them in some of the surfaces it not a problem, as they still would be represented by others. Only for datasets with moving sources this can introduce inaccuracies when the source positions have to be searched and connected through time.

A drawback of this method is, that virtual sources can be extracted only within the domain of the dataset. Virtual sources that are outside of the dataset boundaries are obviously not

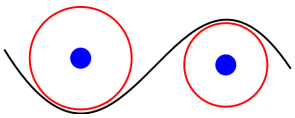
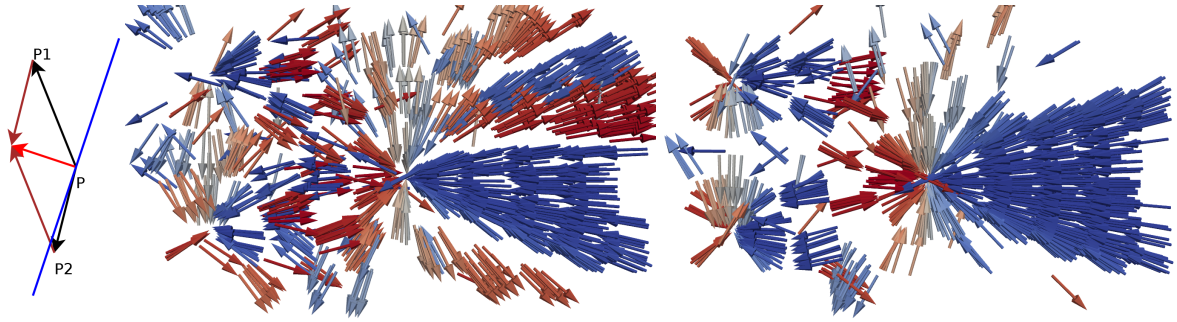


**Figure 5.1.:** Illustration of center of curvature (blue) along a ridge (black).




**(a)** **(b)** **(c)**

**Figure 5.2.:** Consistent orientation of ridge normal vectors for correct projection. Technique for determining flipping of normal vectors (a), unflipped (b) and flipped (c) normal vectors. Vectors are colored by $x$-axis component.

amenable by local minima of the ridge surfaces. This is a case where a curvature-based approach, as follows in the next Section, is promising.

## 5.2. Extraction by Means of Centers of Curvature

An other approach than local minima search on the surfaces, is to use the curvature of the ridge to project where the respective center of curvature lies. As illustrated in Figure 5.1, the curvature of each point on a ridge can be viewed as part of a circle and therefore a projection to the center of the circle can be done. The radius of the circle is determined as the inverse of the curvature.

With the technique described in Section 4.4 reduced from three dimensions to two dimensions separately for each time step, the result are ridge lines instead of surfaces. It has, however, to be extended for the calculation of the normal vectors that are necessary for curvature computation. This is archived using the technique described in [PS08]. In a two-dimensional field for each point the gradient vector $\vec{g}$ and the Hessian matrix $H$ is needed. Using a matrix constructed from the column vectors $\vec{g}$ and $H \cdot \vec{g}$, the determinant $d = \det(\vec{g}|\vec{Hg})$ can be calculated. Then the normal is obtained as $\vec{N} = \nabla d$. While this must be calculated for the whole field, due to the usage of central differences, only the interpolations at $\vec{g} \cdot \vec{ev} = 0$, as found along the ridge, provide normals that point to the center.

However, the normal gives a good representation for the orientation between a ridge sample point and the center of curvature, but its direction is often pointing outwards, not inwards the

curvature. Therefore the technique illustrated in Figure 5.2 (a) is used. By combining the vectors which point from the sample point to its neighbours, a direction that points inwards the curvature (illustrated as cells from P to P1 and P2), is obtained (red). While this does not provide a good normal estimation, it is sufficient to decide if the normal vector must be flipped. It is assumed that if the dot product $\langle(\vec{PP_1} + \vec{PP_2}, \vec{N}\rangle < 0$ the normal must be flipped. On deformed ridges due to inappropriate sampling this could cause the normal to be inconsistently oriented. Figure 5.2 (b) and (c) show the resulting normals before and after flipping.

Now the curvature $\kappa$ can be obtained as the derivative of the tangent with respect to a parametrization of the ridge curve, which is obtained as $\vec{T} = (-N_2, N_1)^T$. Our first approach for obtaining the curvature was using tangent differences between neighboring vertices of the ridge as shown in Equation 5.1. The distance at which the point is then moved along the normal is then obtained as $1/k$.

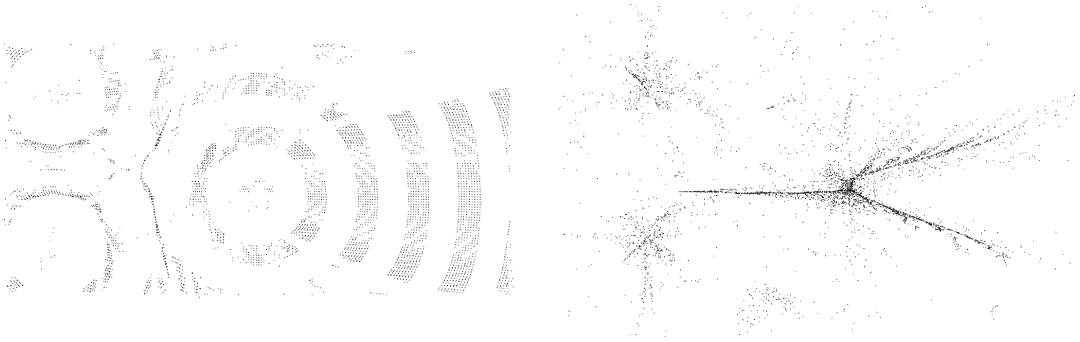
$$\kappa = \frac{\sum_{i}^{Neighbours\ n} \|\vec{T_i} - \vec{T}\|}{n} \tag{5.1}$$

Figure 5.3 shows in (b) the points after projection to the curvature center, from the original points in (a). This however uses a threshold $p_{min}$ for a minimum distance as $(1/\kappa) > p_{min}$, as presumably points that did not substantially move are affected by errors in their calculation. It can be seen, that many points group around the two desired virtual sources, but that also the points scatter very far away from the sources. Outside the three dense regions from sources further dense line-like structures can be observed. Those are typically result from curvature changes due to superpositions, as changing curvature also changes the projection distance. To reduce the scattering, other ways of computing the curvature and the distance of the center of projection were examined.

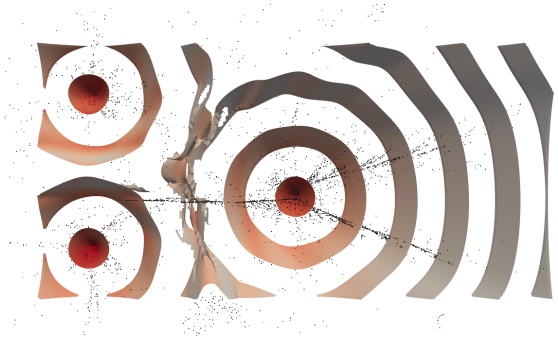
The second approach is using sample points along both directions of the tangent of the sample point, instead of its ridge neighbours, as illustrated in Figure 5.4. This was tried with different distances along the tangent, mostly 1/10 cell size. However, in our tests with a simple one-source dataset, where the ideal distance can be precomputed for comparison, it has shown accuracy inferior to the sampling using ridge neighbours.

The third approach for determining the curvature computes the Jacobian $J$ field from the $\vec{N}$ field. With interpolated $J$ and $\vec{n}$, which must be normalized, at the ridge point the curvature can be calculated using $\kappa = \|J \cdot \vec{n}\|$. However, this approach also provided less accurate results in our experiments, compared to the first method.

If the desired virtual sources are moving, the projected points have to be also projected through time for an adequate space-time visualization. To reflect the time that a wave needs to travel a given distance, one requires the phase velocity $v_p$ and the obtained projection distance $d$. The travel time of the wavefront then can be calculated as $t = d/v_p$ and the position of the centers of curvature are corrected by $t$ back in time.



(a) (b)



(c)

**Figure 5.3.:** Ridge points, original positions (a), centers of curvature (b), and with ridge surfaces in comparison (c). In the images is only a slice of the full dataset, with the points further filtered by a minimum projection distance.

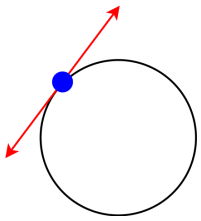


**Figure 5.4.:** Illustration of a curved line (black) and the tangent directions (red) at a point(blue).

For filtering, the curvature parameter on ridge extraction could be used. This is already described in Section 4.4. With higher absolute eigenvalues the strength of the ridge is higher, thus the flatter far away ridges are eliminated. This also is a indication to the curvature within the ridge surface, which is used here for projection. As the distance is calculated as the inverse of the curvature, errors on low curvature values have a high influence on the distance. With the filtering of flat, far away ridges, those error-prone ridge points could be rejected.

As an improvement for the normal flipping, the formulation can be changed back to three-dimensional ridge surfaces. Because the surfaces form funnels around the sources, the surface normal must be pointing upward along the time direction, if the normal points inward the curvature to center. So the normal must flipped if the vector component for the time dimension is negative. As however the normal calculations are good enough, this approach is not used in our implementation.

Due to the huge number of ridge points, the results are so much overlapping that the dense regions could not necessarily be distinguished from scattering. In Figure 5.3 the dense clusters are only clearly visible because only a very thin temporal slice of the whole dataset is displayed. A first approach to improve this was to determine the point density by constructing a scalar field that counts them per cell with a doubled resolution than the original field. This however necessitates further work to extract the cluster centers for further analyses. Another approach for clustering is described in Section 5.3.

## 5.3. Clustering of Extracted Virtual Source Points

As mentioned, the center of curvature projection needs a clustering method for extracting virtual source positions from the set of centers of curvature. Also in case, that the virtual sources are obtained from the minima extraction from the ridge surfaces, they can benefit from clustering methods, to merge the points over time, which represent the same virtual source. To this end, the PCA Split Algorithm 5.1 by Hopf and Ertl [HE03] is used.

The algorithm is based on Principal Component Analysis (PCA). The first step of the algorithm is calculating the center of gravity of the point set and its covariance matrix. Then it must be decided with an error function, if the dataset should be split into two subsets or if the calculated center is a sufficient representation. If not, the principal component is extracted from the covariance matrix and used for splitting the cluster. The eigenvector that represents the principal component is used as a normal vector of a splitting plane. So for each point in the dataset a vector from the dataset center to the point can be used together with the principal component in a dot product. The points are then sorted regarding if the dot product is above zero or not.

For the point extraction methods used in this work, the error function could be based on the calculated covariances and the sample distances $s_{dist}$. On the minima extraction as described in Section 5.1 the criterion $\sqrt{(cov_x^2 + cov_y^2} > s_{dist}$ was used on a two-dimensional PCA-Split, ignoring the time dimension for clustering sources that are not moving.

**Algorithm 5.1** PCA Split Algorithm

```
procedure PCASplit(PointList pl)
    Point avg = CalculateAverage(in pl)                    // Center of cluster
    Matrix cov = CalculateCovarance(in pl)
    if CheckError(in avg, in cov) then
        return avg                          // Error of cluster low enough, return cluster
    else                                          // Error of cluster to high, split cluster
        float[] eigenValues = CalculateEigenvalues(in cov)
        SortEigenvaluesDescending(inout eigenValues)
        Vector principalVector = CalculateEigenvector(in cov, in eigenValues.first)
        Pointlist left, right
        for all Point p ∈ pl do
            Vector positionVector = p − avg
            if DotProduct(principalVector,positionVector) > 0 then
                InsertPoint(inout left, in p)
            else
                InsertPoint(inout right, in p)
            end if
        end for
        PCASplit(in left)
        PCASplit(in right)
    end if
end procedure
```

After clustering the count of points in a cluster can be used as an filtering criterion. Clusters that contain few points are likely caused by superpositions or other errors and can be rejected.

## 5.4. Comparison

In a direct comparison, the precision of the approach based on local minima extraction is higher, as no further error-prone calculations are needed. While the normal calculation in the center of curvature method produces good results, the curvature calculation errors have strong influence on the distance and time offset calculations. However, clustering and filtering by cluster size can deliver acceptable results, but they are typically not as exact as those obtained by the method based on the local minima of the ridge surfaces.

The center of curvature method has an advantage over the local minima extraction in terms of sources that do not lie within the domain of the dataset. The center of curvature method can detect those virtual sources, while obviously the local minimum extraction cannot.

It must be remarked that not all found virtual sources necessarily represent real sources in the dataset. Deformations in curvature from superpositions can cause dense regions in center of curvature projection or local minima in the ridges, as well as reflections of waves. Also, the

clustering methods cannot detect sources that are moving or are heavily distorted through superpositions.

# 6. Virtual Source Signal Reconstruction

The virtual sources obtained so far can be visualized by their geometry only, as points or curves in space-time. Such a visualization provides information where the most important contributions, with respect to the structures which emerge from the positions of the points, come from and how these structures arise from the original field. The motivation for the technique described in this section is to provide additional information about the contributions in terms of field strength. This is achieved in terms of field reconstruction, by trying to determine values along the space-time representation of the virtual sources such that their superposition in terms of sources, including the phase velocity at which their information spreads through the domain, results as close as possible to the original field.

Our model for reconstructing the values of the virtual sources is formulated as follows:

$$S(x) = \sum_{i}^{sources} \frac{\sigma_{i,t-\Delta t}}{(\Delta x_i)^2}, \Delta t_i = \frac{\Delta x_i}{v_p} \tag{6.1}$$

The value $S(x)$ at each point $x$ within the domain of the field shall equal the sum of the influences of sources. The values of the virtual sources are discretized with a uniform temporal sampling, resulting in a set of values $\sigma_{i,t}$ for each source $i$ and discretized time $t$. Also the travel time $\Delta t_i$ of the wave from the respective source $i$ to the point $x$, must be included in the calculation, using the phase velocity $v_p$ and the distance $\Delta x_i$. Thus, for a sample point $x$ at time $t$ the signal at the virtual source $i$ must be fetched at $t - \Delta t$ to accommodate for travel time.

To calculate the virtual source signals $\sigma_{i,t}$ from the given field data the system of equations

$$\begin{array}{l} sample_1\{ \\ sample_2\{ \\ sample_3\{ \\ sample_4\{ \end{array} \left( \begin{array}{cccccc} 0 & d_{11} & 0 & 0 & 0 & d_{12} \\ d_{2,1} & 0 & 0 & d_{2,2} & 0 & 0 \\ 0 & 0 & d_{3,1} & 0 & d_{3,2} & 0 \\ d_{4,1} & 0 & 0 & 0 & 0 & d_{4,4} \end{array} \right) \times \left( \begin{array}{c} \sigma_{1,t_0} \\ \cdots \\ \sigma_{1,t_n} \\ \sigma_{2,t_0} \\ \cdots \\ \sigma_{2,t_n} \end{array} \right) = \left( \begin{array}{c} S(x_1) \\ S(x_2) \\ S(x_3) \\ S(x_4) \end{array} \right) \tag{6.2}$$

$$\underbrace{\qquad\qquad}_{source_1 t_{0..n}} \quad \underbrace{\qquad\qquad}_{source_2 t_{0..n}}$$

must be solved. The matrix has a row per sample point and a column per source and time step. The matrix consists of coefficients $d_{i,j}$ that represent the distance falloff factor from source $j$ to sample $i$ and are only non-zero in the columns, that represents the time at which the signal must have started to arrive at the right time at the sample. The vector which is

multiplied by the matrix represents the unknown source coefficients, for which the system of equations must be solved using the given sample values $S(x)$ on the right hand side. For a robust result it is necessary to overdetermine the system , i.e., more sample points in the field domain are needed than unknown signal coefficients.

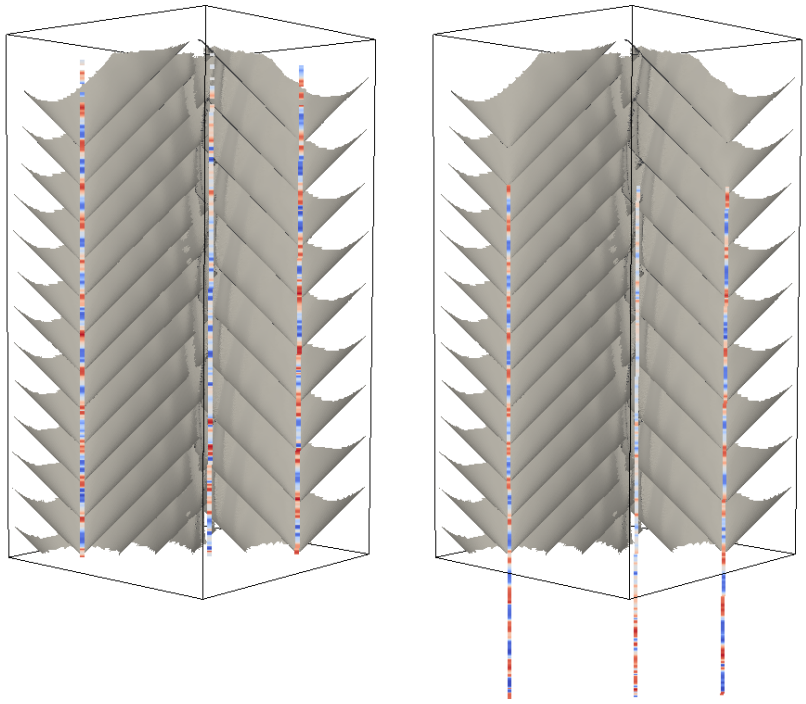
The result of the reconstruction depends, beside other factors, on the quality of the virtual source extraction result from the previous chapter. For these reasons it is likely that the overdetermined system of equations has no exact solution and hence has to be approximated. Therefore the least squares method is used, which tries to find a solution by minimizing the errors with respect to an Euclidean error metric. As the transformation of the equation $A \times x = b$ to $x = A^{-1} \times b$ would require the calculation of the inverse matrix, it is cheaper to use a matrix decomposition method. For this the Cholesky decomposition method [Bjö, p.44] is used. However, Cholesky decomposition requires the matrix to be a symmetric square matrix, which does not directly match the need to overdetermine the system of equations. Hence, Equation 6.2 must be transformed to $A^T \times A \times x = A^T \times b$ prior to the Cholesky decomposition.

Depending on the choice of the sample points $x_i$ it can happen that some virtual source time steps $\sigma_{i,t}$ are never hit by a sample. This would cause that the column and row which represent this time step are all zero and therefore cause the decomposition to fail, because the resulting matrix $A^T \times A$ would be non-invertible. To fix these cases, all zero coefficients in the diagonal of the matrix are set to non zero values and the respective source coefficients are marked as invalid for the result. The respective values at the virtual sources are in our implementation discarded and cause holes in the space-time-curve of the respective virtual source.

The occurrence of such errors and the quality of the reconstructed signal could be optimized by a dedicated method for sample selection. Good results were achieved in our experiments by using a constant number of samples per time slice and randomly choosing x/y positions, in opposition to a fully random sampling within the space-time domain.

Another approach to guarantee that each source coefficient is hit by at least one sample is by back tracing from those, i.e., by starting at each virtual source signal sample and determining a time and location along the propagation through space and time. In doing so the samples were chosen by projecting from each source coefficient into the dataset to choose a sample point, which is then used for signal reconstruction. However, it needs much effort to ensure a good sample distribution in the dataset using this technique, as relating to the source coefficient position and time it can be hard to find samples within the dataset space-time domain. For some source coefficients this approach may not even provide possible sample points if they lie too near at the dataset boundaries and could not be hit by sample points within the dataset.

In contrast to ensuring that every source coefficient is hit by at least one sample, another problem is that for an optimal solution every sample should hit every source exactly once for providing a optimal distribution of the sample value to all superpositioning sources. Depending on how virtual source points from the previous extraction are connected over time or within which time interval the virtual source signal should be reconstructed, it is possible that samples do not hit all sources. This leads to errors which cause a deviation of the result the more, the fewer sources are hit.



(a) (b)

**Figure 6.1.:** Comparison of signal reconstruction for stationary sources with dataset time range (a) and shifted time range (b). The surfaces are only ridges.

As the matrices are very large depending on the number of samples, sources, and reconstructed time steps, the size of the data could be a problem. However, the matrix consists of mostly zero entries. The factor of nonzero values is in optimum at one per number of time steps per source. Mostly it is even a considerably lower factor, as not every source is hit by every sample. Therefore, sparse matrix representation as mentioned in Section 2.5 is of substantial benefit, reducing the memory footprint significantly.

In the following are different approaches described, which were explored in this work. They use different ways of clustering the extracted virtual source points over time and different ways of choosing the time frame for source reconstruction.

## 6.1. Spatially Clustered Stationary Virtual Sources

A first approach was to cluster the extracted source points based on two-dimensional space and to ignore the time dimension, based on the PCA-Split approach with only low error tolerance. This clusters tight virtual source point clusters, as those resulting from ridge surface funnels of

a stationary source, together, but keeps points of distorted or moving sources and other broad clusters apart.

The time span and sampling rate for which the source signals are reconstructed is the same as in the original dataset. However, in a straightforward approach many samples at the lower end of the time domain would be too early to hit virtual sources and the latest source time steps would never be hit by samples at the upper end of the time domain. To accommodate for this the signal reconstruction time span is shifted back in time based on the spatial dataset extent and phase velocity to gain a better sample/source hit ratio. This is done by shifting by $\Delta t = \delta_{\max}/v_p \cdot 1/2$, with $\delta_{\max}$ being the maximum spatial diameter of the dataset, as the mean signal travel time is the relevant factor. The difference can be seen in Figure 6.1 with (a) dataset time span and (b) shifted time span. In (a) the top region exhibit missing values, as no samples have hit them. In (b) the range is complete, but has distortions on the bottom where the number of hits is lower. In both versions in Figure 6.1 it can be seen that the ridges and valleys are correctly hit on the virtual sources curves and that the false source in the middle is distorted and exhibits mostly weaker values.

A problem with this method is that each source requires many time steps to be computed. In this work we used a number of 200 time steps per virtual source. Hence, when many false positive sources exist, for example due to superpositions, the matrix can reach sizes of thousands of columns. On the test hardware the decomposition of up to 40 sources took around 10 minutes, but for 80 sources already two hours. Therefore, an iterative decomposition method was tried in replacement for the Cholesky decomposition. But this did not provide better timings, as it requires many iterations to converge.

## 6.2. Ungrouped Virtual Source Points Local Time Extrapolation

The second approach targets on reducing the sampled time range per source for lower computing costs. In this approach the virtual source points that result from the approach based on minima of the ridge surfaces (Section 5.1) are not further clustered. Each virtual source point is sampled for a fixed time range before and after the point, hence assuming that the virtual point sample can be approximated as a stationary point over this short period of time. With the range set to a suitable value, stationary sources are covered over the full time range, as they are covered by such a time interval from each ridge and valley. Distorted and moving sources are covered locally per ridge and errors from superpositions have only local impact instead of the full extent of the time domain of the dataset. However, if the local time range is chosen too short, stationary sources will exhibit gaps. If, on the other hand, it is chosen too long, overlaps can cause inconsistent results.

Figure 6.2 shows the difference between this approach and the clustered full time sampling of sources. Both original sources that were used to generate the dataset are almost completely captured even with the local sampling method. The described problems of temporal sampling length can be seen, as the left source has minimal overlaps, but the right one exhibits many gaps. In the center the difference to sources that arise from ridge surface errors due to superpositions can be seen. While in global clustering they group together to one virtual source, they now

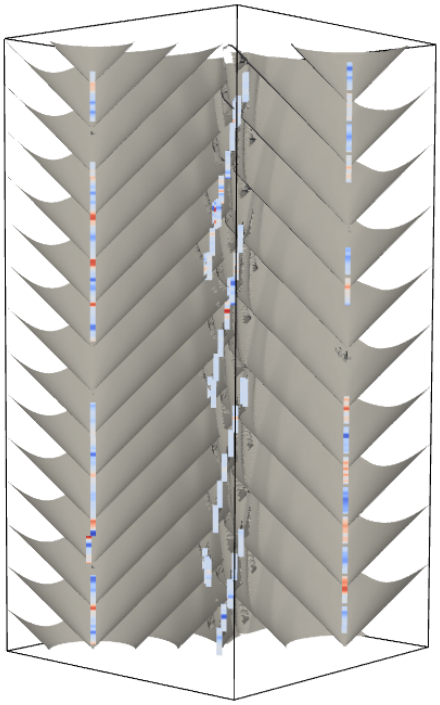


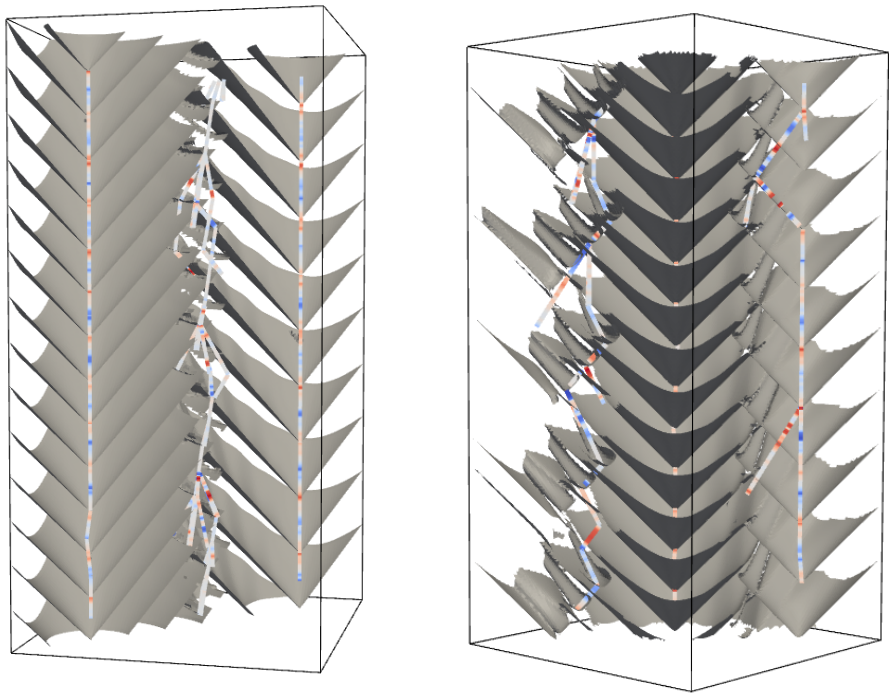
**Figure 6.2.:** Same dataset as in Figure 6.1 but with the local time sampling scheme of the virtual sources. The local minimum extraction errors are scattered to many short time sequences instead of one clustered source.

split to many, even parallel, virtual sources. This lowers the obtained signal values of the original sources, as the sample point values are distributed to much more virtual sources.

## 6.3. Connected Virtual Source Point Interpolation

Our third approach aims at building a connectivity between the extracted points and using the resulting lines for source signal interpolation. The goal is to provide connected space-time curves for moving sources.

The basic strategy of this approach is the clustering by searching from each point a connection to a point later in time. This results in a maximum of one line segment starting per point, but possibly multiple line segments merging into a point. For possible point connections the condition is enforced, that the movement speed between the two end points must be below the phase velocity $v_p$ as $\frac{\Delta spatial}{\Delta time} < v_p$. This origins in the requirement, that our virtual sources always move slower than the phase velocity of their signals waves. This assumption is motivated by the fact that our virtual source visualization is interpreted in terms of waves that travel from these to a location of interest. To choose a point if multiple candidates apply,



**(a)** **(b)**

**Figure 6.3.:** Examples for the connected source point interpolation approach. (a) shows the dataset from Figure 6.1 slightly rotated for a view on the result of the scattered points in the middle. (b) shows a new dataset obtained from three stationary original sources, where the left one is much weaker and strongly distorted.

which meet the maximum motion speed condition, a condition to chose the point with the shortest distance is enforced. As this distance is a space-time distance, the dimensions must be weighted. Therefore distance calculation is calculated as $\sqrt{x^2 + y^2 + (t \cdot v_p)^2}$. Another possible condition for choosing the upper (later) point for a segment would be to use the slowest moving one instead of the above distance measure.

In the previous approaches from each sample only a ray to the spatial position of the source had to be cast and the right time had to be calculated, as detailed above. The procedure is more complex in this approach, as the virtual source line segments can exhibit any orientation in space-time and not only be aligned to the time axis. We computed the crossings according to Equation 6.4, where the left hand side represents the source line segment $Seg$ and the right hand side an upside down funnel originating in the sample point $S$. The Equations

$$x^2 + y^2 = 1 \qquad r > 0 \qquad 0 \leq t \leq 1 \tag{6.3}$$

provide further conditions for obtaining the results, as the radius $r$ of the funnel must be greater than zero and $x$, $y$ give then the direction to the source. $t$ is the parameter for the crossing at the source line segment.

$$\overrightarrow{Seg_{start}} + t \cdot \overrightarrow{Seg_{direction}} = \overrightarrow{S} + r \cdot \begin{pmatrix} x \\ y \\ -1/v_p \end{pmatrix} \tag{6.4}$$

Figure 6.3 shows two examples for this approach. (a) shows the previously used dataset. The original sources are well represented, but the points from inappropriate filtering in the middle show erroneous connections. A fitting example for the described approach is given in (b). The left source is weaker and strongly affected, but at least found and connected. On the right source a deviation in the upper part could be seen. This happens due to the greedy search, as the next both ridge and valley minimum points are missing and the erroneous point has a smaller distance than the next correct local minimum.

On the lower middle part of the right source in (b) an incoming branch from an erroneous point can be seen. A possible approach for filtering such branches would be to search in a set of connected source segments for the longest connected line and crop shorter branches away, before the signal reconstruction is done. Those branches otherwise get hits from dataset samples where only one source segment should be and deteriorate the result.

## 6.4. Comparison

Comparing the spatial clustering method from Section 6.1 to the local extrapolation method described in Section 6.2, the first provides smoother results, as overlaps and gaps are avoided. Parallel erroneous segments as often results due to superpositions can be substantially reduced by clustering. With the local extrapolation approach however, moving sources are much better represented, as they are calculated locally and are not approximated by a rough cluster center. The greedy connected interpolation approach presented in Section 6.3 produces the best results for stationary and moving sources. However, erroneous points and distorted sources result also here in erroneous connections. At least, the local minimum source extraction as in Section 5.1 can reduce the number of erroneous points by filtering. In the dataset used in this chapter fringe cutting was deactivated for demonstration purposes. For datasets with weak and strongly distorted stationary sources as the left one in Figure 6.3 (b), spatial clustering for stationary sources should provide the best results through the averaging from clustering. The connected interpolation method, however, still finds and connects them.

The necessary computing power is highly dependent on the clustering and filtering. With inappropriate filtering and low error tolerance clustering, the first method produces many sources with many samples per source, while the local clustering employs much fewer samples per source. The greedily connected points in this case should have the medium to highest computational cost, as a result from more complex sample calculations. The number of source

samples over all is of major impact here, as the signal reconstruction depends on the Cholesky decomposition, which has complexity $O(n^3)$.

An important point which must be considered is the quality of the reconstructed signal. The results are, if real sources are captured by our virtual source approach, typically similar to the original signal which was used to build the dataset. On the boundaries of the datasets the calculated signals exhibit often strong outliers, and on the interior the resulting values are typically much lower than expected. This depends on the quality of the data samples and how often each source sample is hit. More hits smoothen the result, what explains the outliers on the boundaries, as the hit probability is lower there. Dataset samples should ideally hit each original source once. Due to the clustering methods and time ranges, not every source is hit by a sample. This makes the value distribution of the sample less continuous. The other effect is that additional sources from erroneous points, which distribute the sample values to more sources than actually exist, significantly lower the reconstructed signal values.

It should be also noted that in few cases, most commonly in the case of the local reconstruction approach, as in Section 6.2, the Cholesky decomposition was not successful due to an "numerical issues" error. It is unclear whether this originates from degenerated rows of the least squares matrix due to inappropriate samples or from a problem within Eigen, the linear algebra library which is used for these calculations. However, if the decomposition does not abort due to error, which is the typical case, the results are as expected.

# 7. Results

We applied the introduced techniques to different datasets that were generated using the software which is described in Chapter 3 and to an electromagnetic field simulation, which bases on the finite-difference time-domain method and is also mentioned in the respective chapter.

Figure 7.1 shows a result for a more complex dataset. The dataset was constructed by two sources of same frequency, visible on the right, and one with half that frequency on the left. As indicated by the ridge and valley surfaces of the wavefronts at the bottom, a full surface visualization would suffer from clutter, making it hard to identify the inherent structure. It also can be seen that at the boundary between the two sources to the left many artifacts persist after filtering. This results a tree structure due to the greedy virtual source connection, with a less regular signal reconstruction. The other virtual sources which represent real sources of the dataset exhibit a regular signal reconstruction, on which the different frequencies can be identified. However, on the left, lower frequency, virtual source some points are missing from the local minima extraction. Therefore the uppermost virtual source point connects to a virtual source point of the artifacts between the original sources.

A second dataset which was generated using the finite-difference time-domain method (implementation by Thomas Müller) is visualized in Figure 7.2. The dataset contains a stationary oscillating source and a reflective boundary condition on the left. Beginning at the bottom, it can be seen when the first wavefront hits the reflective boundary and is reflected back into the field domain. It can be seen how reflected wavefronts superimpose the original wavefronts. Examining the red dots, which represent the spatial center of curvature projections, on the right side of the dataset dense lines could be seen. Due to varying curvature of the ridges, they form lines instead of point shaped dense regions. The rather circular regions of points in the middle of the domain are even more dense, but this is occluded by the ridge surfaces. It can further be seen that the lateral lines are not as clearly visible on the left side of the dataset, as the reflections cause further spread on this side.

## 7.1. Timings

Providing that the virtual sources are appropriately filtered, computation time is not an issue for the approach presented in this work. However, as the current implementation for this work precomputes and stores the gradient vector, the Hessian matrix and an eigenvector at each node of the computation grid for the ridge extraction, system memory consumption is a problem. Including the original scalar field, 16 double values are stored per dataset node.

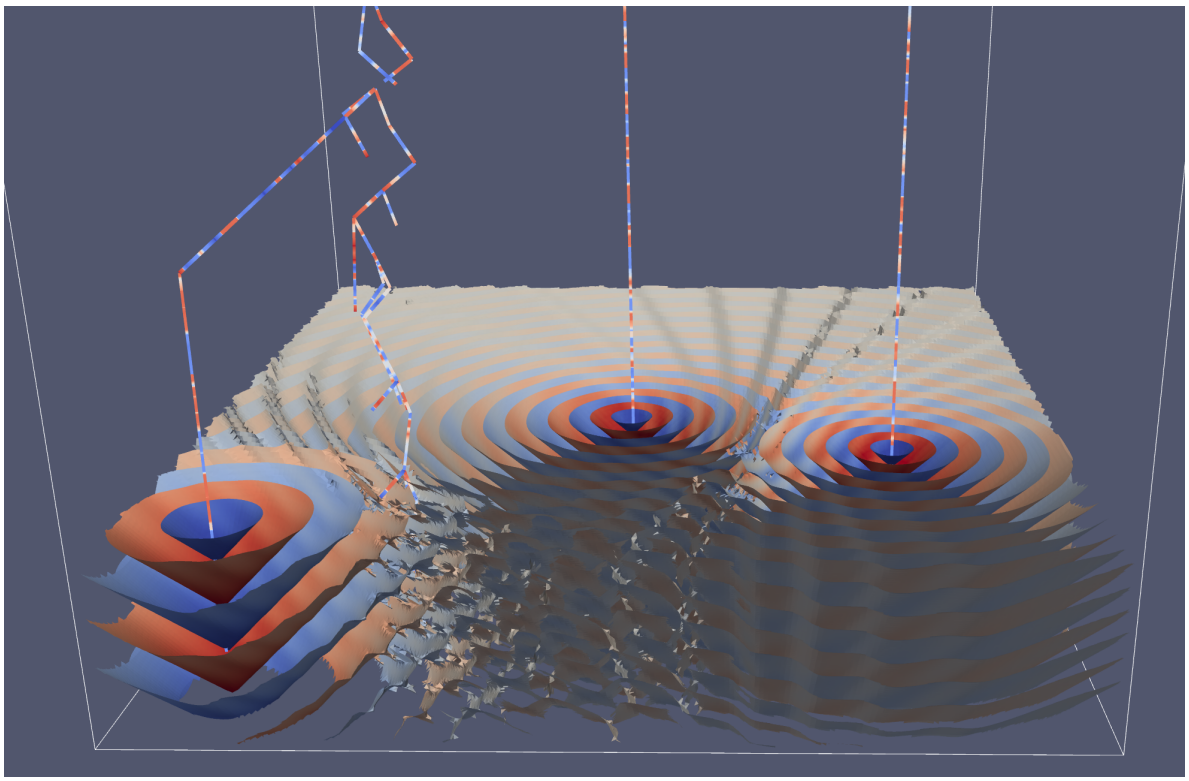


**Figure 7.1.:** Dataset consisting of three sources with different frequencies with greedy connected virtual source points. They visualize the reconstructed signals, providing a notion of the field.

This results in a memory consumption of ∼122MB for a $100^3$ cube dataset, still without the extracted features. Table 7.1 shows the timings for two datasets. The first dataset consists of $100 \times 100 \times 200$ points and is shown in Figure 6.3 (b). The second dataset two has a resolution of $200 \times 200 \times 400$ and is shown in Figure 7.1.

The timings of the signal reconstruction can be interpreted by the expenses of dataset sampling. As 200 samples are taken per time slice on the runs in this table, they have a strong influence in comparison to the resulting virtual source signal samples. Stationary sources involve the simplest sample calculations and are therefore typically the fastest, as only the distance between dataset sample and virtual source sample must be calculated for obtaining the travel time of the wave. The local sampling includes further range checks on the time axis, but is still simple. Only the calculations for the greedy connected virtual sources require the expensive calculation of a crossing point between a funnel from the dataset sample and a line from the virtual source segment.

[1]Triangle count pre Filtering, 2 fringe cut iterations
[2]4-Neighbourhood search range
[3]From local minima Points

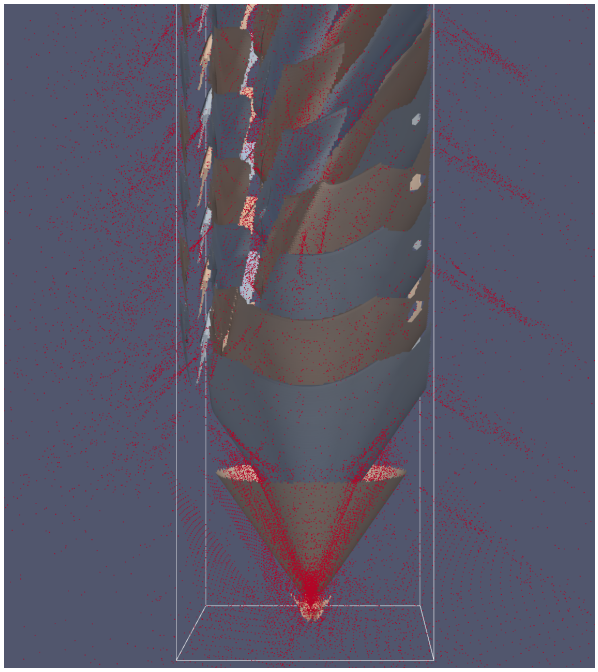


**Figure 7.2.:** Electromagnetic field dataset simulated using the finite-difference time-domain method. A stationary oscillating source located at the spatial center with a reflective boundary condition on the left. The red points visualize the centers of curvature obtained according to the technique described in Section 5.2. Time increases in upward direction while the spatial dimensions are horizontal.

| Step | Dataset 1 | | Dataset 2 | |
|---|---|---|---|---|
| Ridges[1] | 639471 Triangles | 8.81 | 7576767 Triangles | 99.23 |
| Valleys[1] | 637974 Triangles | 8.74 | 7538483 Triangles | 100.35 |
| Local minima Sources[2] | 63 Points | 2.56 | 242 Points | 37.15 |
| Center of Curvature Sources | 42112 Points | 2.02 | 288622 Points | 18.32 |
| PCA Split for minima Points | 4 Points | <0.01 | 4 Points | <0.01 |
| PCA Split for curvature Points | 54 Points | 0.02 | 183 Points | 0.30 |
| Signal reconstruction[3] | | | | |
| - stationary | 800 Samples | 0.10 | 1600 Samples | 0.70 |
| - local | 630 Samples | 0.33 | 2420 Samples | 6.18 |
| - greedy connected | 707 Samples | 0.64 | 1626 Samples | 5.47 |

**Table 7.1.:** Computation times in seconds.

# 8. Conclusion

In this work we have introduced methods based on space-time height ridge extraction for the visualization of wave propagation phenomena and introduced feature extraction techniques, that are based on the extracted space-time wavefront surfaces, to provide a concise visualization. With the introduction of virtual sources and signal reconstruction based thereon we have provided a visualization technique with reduced occlusion and complexity, as compared to the visualization by volume rendering or ridge surfaces.

We presented two different approaches for the extraction of virtual sources. One approach based on local minimum extraction within the obtained ridge surfaces, the other based on a projection to the spatial center of curvature of the space-time ridge surfaces. It must, however, be noted, that the results from source extraction based on curvature projection result in many more virtual sources due to curvature variation of the ridges. Hence, this approach does not provide as robust results and requires special attention on parameter fitting for clustering. In contrast, the results based on the local minimum extraction do not exhibit this drawback, as long as the sampling rates are high enough. This is important as for the subsequent signal reconstruction at the virtual sources in space-time the results can only be as good as the preceding virtual source extraction step.

In our signal reconstruction the results show a good representation of when height ridges or valleys have started at the respective virtual sources, as long as the space-time-curves are constructed from an adequately filtered virtual source point extraction. The resulting signals however are typically jittered, in dependence on spatial and temporal sampling. Nevertheless, although an exact quantitative reconstruction of the original field is typically not feasible at moderate computation times, the obtained results serve well for visualization purposes. There are, however, several potentials for future improvements by more sophisticated filtering, e.g., based on the regularity of the reconstructed signal, and advanced space-time sampling techniques for the signal reconstruction step.

## 8.1. Future Work

There are different branches in which further work could be done. As the ridge surface extraction and the local minimum extraction deliver adequate results, the curvature projection stage and signal reconstruction step are particularly eligible.

A first point to improve could be a extension of the virtual source connection algorithm from Section 6.3. Branches could be cut by searching the longest path of connections and cropping all shorter branches off. This is assumed to improve the quality of the signal reconstruction

substantially. Also the greedy connection search itself could be improved by a better decision between near points and fast virtual source movement or distant points and slow virtual source movement. Thereby, the algorithm must cope with the relation between spatial distance and temporal distance of virtual source points.

Another point to improve would be the data sampling for signal reconstruction. More sophisticated selection algorithms, in contrast to the random sampling used in this work, are likely to improve the signal reconstruction at the virtual sources. This could be achieved by imposing additional constraints on the samples. A good sample should hit many different virtual sources for a smooth result, and signal samples should be hit often for a good reconstruction.

On the curvature projection approach for virtual source extraction, the normal calculations deliver good results. However, the curvature calculation and therefore the projection distance scatter substantially. While the virtual sources can be identified as clusters in the centers of curvature, they are typically not accurate enough for a smooth source signal reconstruction. Therefore, the curvature calculation would be a significant branch for future work for improving the results of the virtual source extraction step. This is furthermore significant, as the local minimum based virtual source extraction is limited to the dataset domain, and hence unable to extract virtual sources outside of the dataset, as for example are expected to appear due to reflection.

# A. Implementation

The tools and visualization methods which are developed in this work are based on the Visualisation Toolkit (VTK). This framework enables the usage of many available visualization methods and is including a rendering framework. As the developed methods are implemented as custom filters, they can be freely combined. The framework also offers the infrastructure for the handling of datasets themselves, which enables the output of interim results between filtering methods at no additional development cost.

The projects were developed in C++ and use Visual Studio 2012, thus some elements in the source code require C++11 compiler compatibility.

## A.1. Dependencies

### VTK 5.10.0

As stated above, the tools and filters are implemented within VTK and therefore depend on it. For the dataset generation tools VTK must be build with QT Support, as the QVTK Widget is used in the user interface.

### QT Libs 4.8.3

The dataset generation tools use QT for the user interface. In the filter project QT is only used for the file dialog to choose a dataset. The filters itself do not use QT.

### Eigen 3.1.1

Eigen is used for sparse matrix decomposition in the context of the signal reconstruction filters only.

### Linalg

A library for two- to four-dimensional vector and matrix calculations, which was written by Ronald Peikert, is used for simple linear algebra problems.

## A.2. Overview

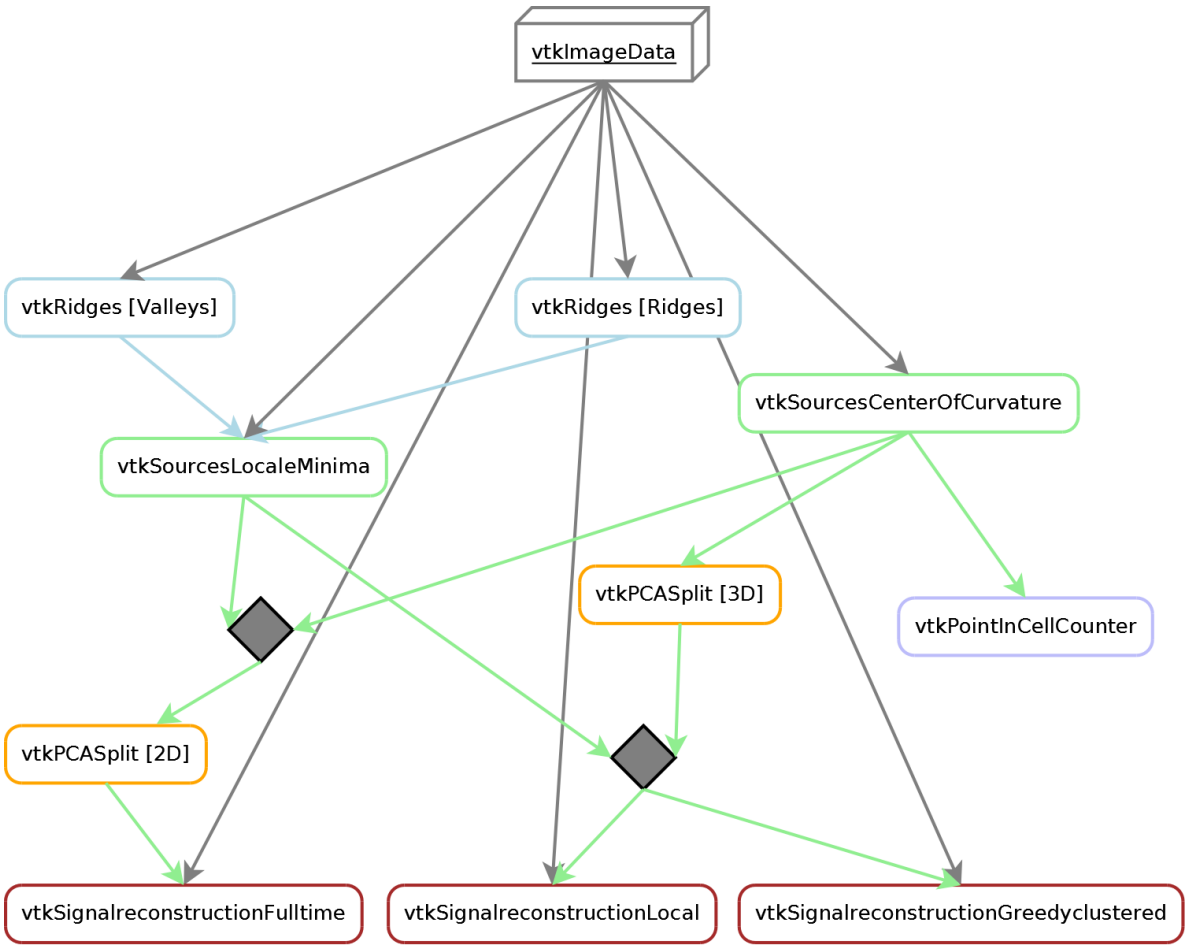



**Figure A.1.:** Overview of possible filter combinations. Alternative choices are indicated by a rue.

The custom filters depend on different input types and should be connected via the VTK pipeline by means of their input and output ports. The initial input is a vtkImageData object with scalar field values. As the datasets from Section 3.2 have two three-dimensional scalar values, one of them must be chosen and prepared with the vtkImageMagnitude filter to derive the respective scalar field for visualization. Figure A.1 shows all possible and useful combinations of the implemented custom filters.

There exists no explicit filter for the gradient magnitude method based on isosurface extraction, which is described in Section 4.3. The method can be realized by using the build-in VTK filters vtkGradientFilter, vtkImageMagnitude, and vtkContourFilter in the mentioned order.

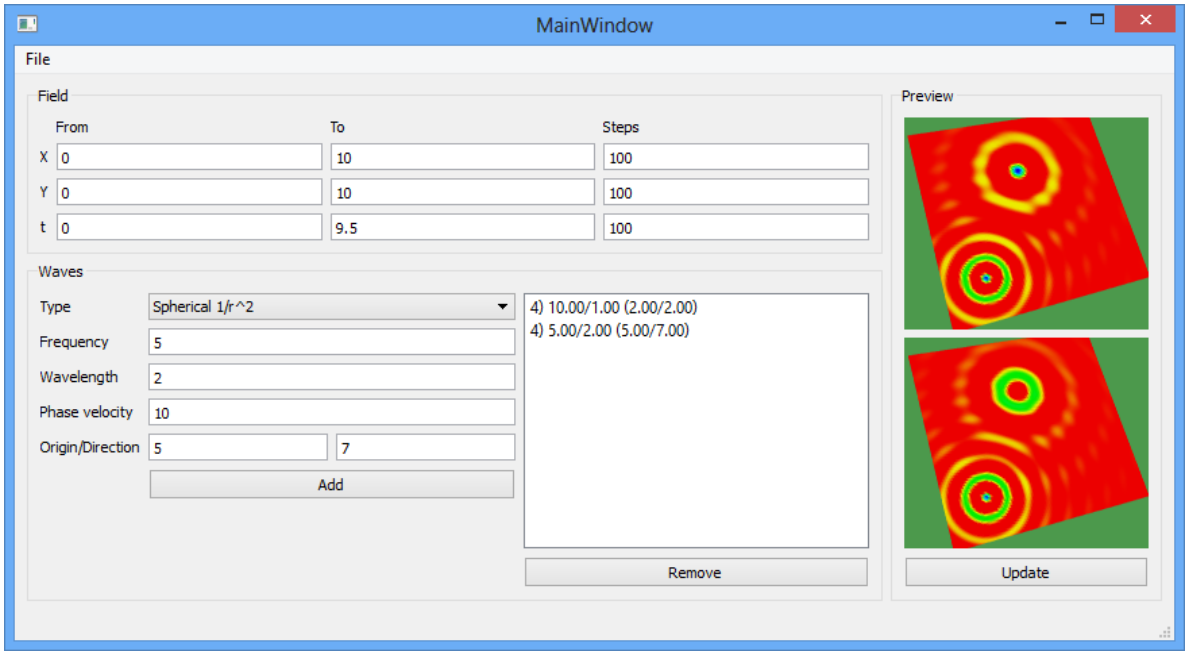



**Figure A.2.:** Scalar field creator GUI

## A.3. Tool: Scalar Field Creator

### Description

The tool, is used for creating scalar fields with $x$ and $y$ as spatial dimensions and $z$ as time dimension. Sources of different wave equation types can be combined to a resulting scalar field. The used equations are described in Section 3.1.

### Usage

The user interface is shown in Figure A.2. The upper matrix of input fields determines the bounds and sampling rates for the output scalar field. To the right are previews of lowest and top time slices, which must be updated manually by the update button.

For a wave, inputs for frequency in $1/s$ and wavelength in $m$ must be given. Phase velocity is calculated by the interface and shown in $m/s$. Depending on the wave type, the input vector is the source origin or the wave direction. However, the list in the middle shows the added sources with the data as $wave\ type) frequency / wavelength (vector_x / vector_y)$. Sources can be selected in the list and be removed.

The output is created by using the menu $file \rightarrow create\ vti$ which opens a file save dialog.

### Output

Three-dimensional vtkImageData. $x$ and $y$ dimensions are spatial dimensions and scaled in meters. $z$ dimension is a time dimension and scaled in seconds. Data is assigned per node of the grid and contains a one-dimensional scalar value of type double.

## A.4. Tool: Dipole E-M-Field Creator

### Description

The tool is used for creating electromagnetic vector fields with $x$ and $y$ as spatial dimensions and $z$ as time dimension. It is used to place multiple Hertzian Dipoles in space. However, they are all aligned along the $z$ axis, standing vertical on the $x/y$ plane. The used equations are described in Section 3.2.

### Usage

The user interface is shown in Figure A.3. The upper matrix of input fields determines the output dataset bounds and sampling rates. To the right are previews of the lowest and the top time slice, which must be updated manually by the update button.

For a electromagnetic dipole, inputs for frequency in $1/s$, dipole moment in $F/m$ and zero phase in $rad$ must be given. The origin determines the position of the dipole. The list in the middle shows the added sources with the data as $frequency\ zero\ phase/dipole\ moment(origin_x/origin_y)$. Sources can be selected in the list and can be removed.

### Output

Three-dimensional vtkImageData. $x$ and $y$ dimensions are spatial dimensions and scaled in meters. $z$ dimension is a time dimension and scaled in seconds. Data is assigned per node of the grid and contains two thee-dimensional vectors of type double.

## A.5. Filter: vtkRidges

### Description

Extracts ridge or valley surfaces as described in Section 4.4 using our marching ridges algorithm.

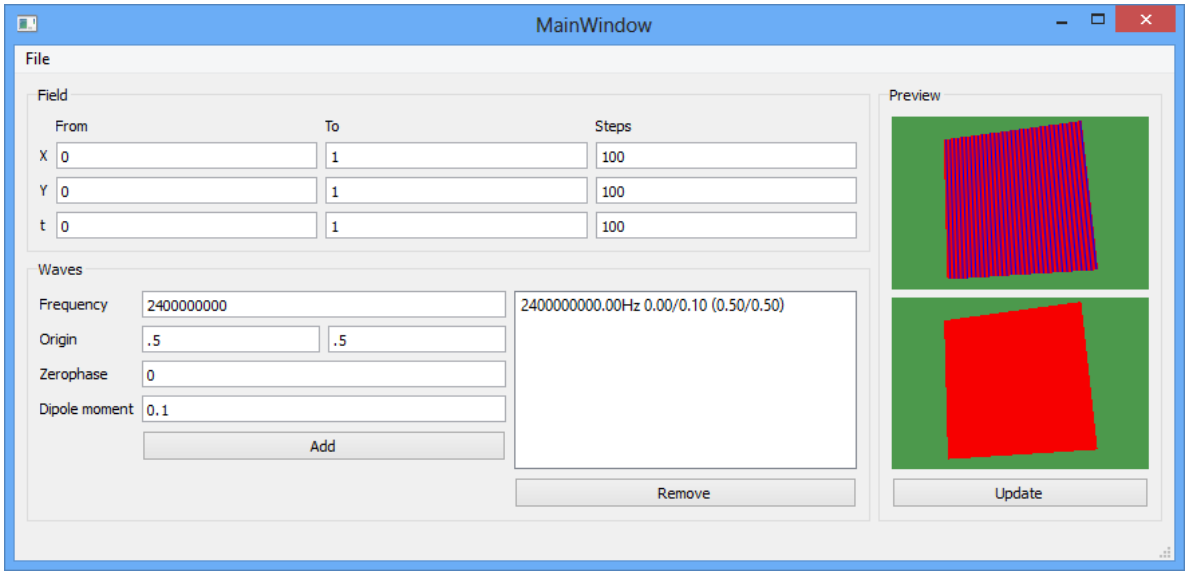



**Figure A.3.:** Dipole E-M-Field Creator GUI

### Input

**Port 0: Base Scalar field** Three-dimensional vtkImageData. The scaling of the axis is not significant for a correct result. The appended data must contain a scalar per point, which is of type double.

### Output

**Port 0: Extracted Ridge/Valley Surfaces** Type of vtkPolyData with triangles as three points per vtkCell. Vertices are shared on adjacent triangles. Data contains a scalar value which is interpolated from the input field. Scalar is of type double.

### Parameters

**SetExtractRidges()** Sets the extraction method to ridges.

**SetExtractValleys()** Sets the extraction method to valleys.

**SetFilterMinimumCurvature(double tau)** A filter on the strength of ridge/valley sharpness. Must be greater or equal to zero.

**SetFilterScalarrange(double min, double max)** A filter on the scalar value for a ridge. Max must be greater than min.

**SetFilterMinimumComponentSize(int triangles)** A filter on the number of connected triangles. Connected components with less triangles are discarded.

**SetFilterFringeCutIterations(int i)** A filter for interferences. Triangles, which have at least on edge that is not connected to an other triangle, i.e., reside at the mesh boundary, are discarded. With the parameter $i$ this can be repeated multiple times.

## A.6. Filter: vtkSourcesLocalMinima

### Description

Extracts local minima from ridge and valley surfaces. The base scalar field is needed, for deciding whether a found minimum is on the bottom plane of the dataset extent.

### Input

**Port 0: Base Scalar Field** Three-dimensional vtkImageData. The scaling of the axis is not significant for a correct result.

**Port 1: Ridges** Type of vtkPolyData with triangles as three points per vtkCell. Vertices must be shared between adjacent triangles.

**Port 2: Valleys** Type of vtkPolyData with triangles as three points per vtkCell. Vertices must be shared between adjacent triangles.

### Output

**Port 0: Source Points** Type of vtkPolyData with vertices as one point per vtkCell. Only points which are assigned to a cell are valid. No further data is appended.

### Parameters

**SetFilterNeighbourrange(int range)** A parameter for neighbourhood size in which a point must be the minima to be accepted.

## A.7. Filter: vtkSourcesCenterOfCurvature

### Description

Extracts sources from a scalar field by calculating the centers of curvature from two-dimensional ridge surfaces per time slice. The method is described in Section 5.2.

### Input

**Port 0: Base Scalar Field** Three-dimensional vtkImageData. The $x$ and $y$ axis must be scaled in meters and the $z$ axis in seconds. The appended data must contain a scalar per node of type double.

### Output

**Port 0: Source Points** Type of vtkPolyData with vertices as one point per vtkCell. Only points which are assigned to a cell are valid. The appended data contains a scalar value per point, which contains the curvature and a vector value, which contains the normal vector. Output is of type double.

### Parameters

**SetProjectionForTimeOn(bool b)** Defines if the projection (i.e., the center of curvature computation) should not only be done in the spatial dimensions, but also in time.

**SetPhasevelocity(double pv)** Sets the phase velocity $m/s$ of wave propagation.

**SetNormalflippingOn(bool b)** Set whether the internal method for consistent vector orientation should be used or not.

**SetFilterProjectionDistance(double minDist, double maxDist)** Sets a filter based on the projection distance which is calculated from the curvature.

**SetCurvatureMode(int m)** A switch for internal curvature calculation methods. Mode 0 selects curvature from tangent differences between ridge neighbours. Mode 1 interpolates points at 1/10 of cell size along the point tangent to calculate curvature from the tangent differences. Mode 2 uses a Jacobi matrix. Mode 0 typically delivers best results.

**SetExtractRidges()** Sets the extraction method to ridges.

**SetExtractValleys()** Sets the extraction method to valleys.

**SetFilterMinimumCurvature(double tau)** A filter on the strength of ridge/valley curvatures. Must be greater or equal to zero.

**SetFilterScalarrange(double min, double max)** A filter on the scalar value for a ridge. $max$ must be greater than $min$.

## A.8. Filter: vtkPointInCellCounter

### Description

Lays a grid over a point input and counts the number of points per cell.

### Input

**Port 0: Source Points** Type of vtkPolyData with vertices as one point per vtkCell. Only points which are assigned to a cell are valid.

### Output

**Port 0: Source Point Image** Type of vtkImageData with a scalar per point which contains the number of counted points. The scalar type is double.

### Parameters

**SetOrigin(double point[3 )]** Sets the origin of the resulting image data.

**SetExtent(int extent[6 )]** Sets the extent of the resulting image data.

**SetSpacing(double spacing[3 )]** Sets the spacing between nodes of the resulting image data.

**AddOuterPointsToBorder(bool b)** Sets whether points which lie outside the result dataset bounds should be discarded or added to the dataset boundary cells.

## A.9. Filter: vtkPCASplit

### Description

Implements the PCA Split clustering algorithm as described in Section 5.3. For different purposes the clustering could ignore the $z$ time dimension and only cluster using the spatial dimensions $x$ and $y$. If clustering should use all three dimensions, it may be necessary to set coefficients per dimension for the covariance calculations, as time and spatial dimensions may need different weight factors.

### Input

**Port 0: Source Points** Type of vtkPolyData with vertices as one point per vtkCell. Only points which are assigned to a cell are valid.

### Output

**Port 0: Source Points** Type of vtkPolyData with vertices as one point per vtkCell. Only points which are assigned to a cell are valid.

### Parameters

**SetFilterMinimumCluserSize(unsigned size)** Sets a filter, which discards clusters that contain less than *size* points.

**SetMaxErrorForSplit(double error)** Sets the error value on covariance, on which a set of points is split to two subsets.

**SetDimensionCoefficients(double x, double y, double t)** Sets coefficients for each dimension on the covariance calculation. For the usage in this work typical usage would be $x = 1, y = 1, t = v_p$.

**SetIgnoreTimeDimension(bool b)** Sets whether clustering should be two-dimensional or three-dimensional, including the time dimension.

## A.10. Filter: vtkSignalreconstructionFulltime

### Description

Performs a signal reconstruction as described in Section 6.1. It makes sense to use a two-dimensional PCA-Split clustering beforehand. For a correct result the phase velocity must be set.

### Input

**Port 0: Base Scalar Field** Three-dimensional vtkImageData. The $x$ and $y$ axis must be scaled in meters and the $z$ axis in seconds. The appended data must contain a scalar per point, which is of type double.

**Port 1: Source Points** Type of vtkPolyData with vertices as one point per vtkCell. Only points which are assigned to a cell are valid.

### Output

**Port 0: Source Signal Lines** Type of vtkPolyData with line lists as two or more points per vtkCell. A scalar which contains the calculated signal value is assigned per vertex. Data type is double.

### Parameters

**SetPhasevelocity(double pv)** Sets the phase velocity in $m/s$ of wave propagation.

**SetTimeTranslationFactor(double f)** A internal time span is calculated on base of the maximum distance between a source and dataset edge points with respect to the phase velocity. The factor is a multiple of the duration by which the signal reconstruction is translated back in time. Default is 0.5.

**SetSamplesPerTimeslice(unsigned int i)** Sets the number of samples which are taken per time slice. This must be greater than the number of input sources.

## A.11. Filter: vtkSignalreconstructionLocal

### Description

Performs a signal reconstruction as described in Section 6.2. As from each ridge a point per source should be included, the time range must be configured for minimal overlapping to produce good results. For a correct result the phase velocity must be set.

### Input

**Port 0: Base Scalar Field** Three-dimensional vtkImageData. The $x$ and $y$ axis must be scaled in meters and the $z$ axis in seconds. The appended data must contain a scalar per point of type double.

**Port 1: Source Points** Type of vtkPolyData with vertices as one point per vtkCell. Only points which are assigned to a cell are valid.

### Output

**Port 0: Source Signal Lines** Type of vtkPolyData with line lists as two or more points per vtkCell. A scalar which contains the calculated signal value is assigned per vertex. Data type is double.

### Parameters

**SetPhasevelocity(double pv)** Sets the phase velocity $m/s$ of wave propagation.

**SetCoefficientsPerSource(unsigned int i)** Sets the time range over which the signal for a source point is calculated, as the signal samples are set with the dataset time sampling distance.

**SetSamplesPerTimeslice(unsigned int i)** Sets the number of samples which are taken per time slice. There must be more samples taken, than coefficients for sources are reconstructed.

## A.12. Filter: vtkSignalreconstructionGreedyclustered

### Description

Calculates source signals as described in Section 6.3. For a correct result the phase velocity must be set. For each source line segment the covered time length is divided by the sampling distance of the time axis. If the segment is too long, the number of samples for this segment is limited by SetMaxCoefficientsPerSource.

### Input

**Port 0: Base Scalar Field** Three-dimensional vtkImageData. The $x$ and $y$ axis must be scaled in meters and the $z$ axis in seconds. The appended data must contain a scalar per point of type double.

**Port 1: Source Points** Type of vtkPolyData with vertices as one point per vtkCell. Only points which are assigned to a cell are valid.

### Output

**Port 0: Source Signal Lines** Type of vtkPolyData with line lists as two or more points per vtkCell. A scalar which contains the calculated signal value is assigned per vertex. Data type is double.

### Parameters

**SetPhasevelocity(double pv)** Sets the phase velocity $m/s$ of wave propagation.

**SetMaxCoefficientsPerSource(unsigned int i)** Sets how many samples at maximum are used per source connection.

**SetSamplesPerTimeslice(unsigned int i)** Sets the number of samples which are taken per time slice. There must be more samples taken, than coefficients for sources are reconstructed.

# Bibliography


[BDM+] M. Bertram, E. Deines, J. Mohring, J. Jegorovs, H. Hagen. Phonon Tracing for Auralization and Visualization of Sound. In *IEEE Visualization*, p. 20. IEEE Computer Society. (Cited on page 10)

[Bjö] Å. Björck. *Numerical Methods for Least Square Problems*. Miscellaneous Bks. (Cited on pages 15 and 36)

[BSDW12] S. Bachthaler, F. Sadlo, C. Dachsbacher, D. Weiskopf. Space-Time Visualization of Dynamics in Lagrangian Coherent Structures of Time-Dependent 2D Vector Fields. *International Conference on Information Visualization Theory and Applications*, pp. 573–583, 2012. (Cited on page 11)

[BSW+12] S. Bachthaler, F. Sadlo, R. Weeber, S. Kantorovich, C. Holm, D. Weiskopf. Magnetic Flux Topology of 2D Point Dipoles. *Computer Graphics Forum*, 31(3):955–964, 2012. (Cited on page 10)

[Dam] J. N. Damon. Properties of Ridges and Cores for Two-Dimensional Images. *Journal of Mathematical Imaging and Vision*, (2):163–174. (Cited on page 10)

[DBM+06] E. Deines, M. Bertram, J. Mohring, J. Jegorovs, F. Michel, H. Hagen, G. M. Nielson. Comparative Visualization for Wave-based and Geometric Acoustics. *IEEE Trans. Vis. Comput. Graph.*, 12(5):1173–1180, 2006. (Cited on page 10)

[Dei08] E. Deines. *Acoustic Simulation and Visualization Algorithms*. Ph.D. thesis, 2008. (Cited on page 10)

[Ebe96] D. Eberly. *Ridges In Image And Data Analysis (Computational Imaging And Vision)*. Springer, 1996. (Cited on pages 10 and 22)

[FKMP] J. D. Furst, R. S. Keller, J. Miller, S. M. Pizer. Image Loci are Ridges in Geometric Spaces. In B. M. ter Haar Romeny, L. Florack, J. J. Koenderink, M. A. Viergever, editors, *Scale-Space*, Lecture Notes in Computer Science, pp. 176–187. Springer. (Cited on page 10)

[FP98] J. D. Furst, S. M. Pizer. Marching optimal-parameter ridges: An algorithm to extract shape loci in 3D images. In W. Wells, A. Colchester, S. Delp, editors, *Proceedings of the 1st International COnference on Medical Image Computing and Computer-Assisted Intervention – MICCAI'98*, number 1496 in Lecture Notes in Computer Science, pp. 780–787. Springer-Verlag, M.I.T., Cambridge, MA, U.S.A., 1998. (Cited on page 10)

[FP01] J. D. Furst, S. M. Pizer. Marching ridges. In M. H. Hamza, editor, *SIP*, pp. 22–26. IASTED/ACTA Press, 2001. (Cited on pages 10 and 23)

[GMME12] T. M. D. M. Gustavo M. Machado, Filip Sadlo, T. Ertl. Visualizing Solar Dynamics Data, 2012. Proceedings of International Workshop on Vision, Modeling and Visualization (VMV) 2012, to appear. (Cited on pages 10 and 11)

[HE03] M. Hopf, T. Ertl. Hierarchical Splatting of Scattered Data. In *Proceedings of IEEE Visualization '03*. 2003. (Cited on page 31)

[Kar] K. Kark. *Antennen und Strahlungsfelder: Elektromagnetische Wellen auf Leitungen, im Freiraum und ihre Abstrahlung*. Aus dem Programm Informationstechnik. (Cited on page 18)

[KK05] S.-K. Kim, C.-H. Kim. Finding ridges and valleys in a discrete surface using a modified MLS approximation. *Computer-Aided Design*, 37(14):1533–1542, 2005. (Cited on page 11)

[KSSW09] G. L. Kindlmann, R. San Jose Estepar, S. M. Smith, C.-F. Westin. Sampling and Visualizing Creases with Scale-Space Particles. *IEEE Trans. Visualization and Computer Graphics*, 15(6):1415–1424, 2009. doi:10.1109/TVCG.2009.177. (Cited on page 11)

[KTW06] G. Kindlmann, X. Tricoche, C.-F. Westin. Anisotropy Creases Delineate White Matter Structure in Diffusion Tensor MRI. In *Ninth International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI'06)*, Lecture Notes in Computer Science 4190, pp. 126–133. Copenhagen, Denmark, 2006. (Cited on page 10)

[LC87] W. E. Lorensen, H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *COMPUTER GRAPHICS*, 21(4):163–169, 1987. (Cited on page 22)

[LCM07] C. Lauterbach, A. Chandak, D. Manocha. Interactive sound rendering in complex and dynamic scenes using frustum tracing. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1672–1679, 2007. (Cited on page 10)

[Lin96] T. Lindeberg. Edge Detection and Ridge Detection with Automatic Scale Selection. *International Journal of Computer Vision*, 30:465–470, 1996. (Cited on page 10)

[MOD00] M. Monks, B. M. Oh, J. Dorsey. Audioptimization: Goal-Based Acoustic Design. *IEEE Comput. Graph. Appl.*, 20(3):76–91, 2000. (Cited on page 10)

[OBS04] Y. Ohtake, A. Belyaev, H.-P. Seidel. Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph.*, 23(3):609–612, 2004. (Cited on page 11)

[OMD+] H. Obermaier, J. Mohring, E. Deines, M. Hering-Bertram, H. Hagen. On Mesh-Free Valley Surface Extraction with Application to Low Frequency Sound Simulation. *IEEE Trans. Vis. Comput. Graph.*, (2):270–282. (Cited on page 10)

[OU] A. Omoto, H. Uchida. (Cited on page 10)

[PR99] R. Peikert, M. Roth. The "Parallel Vectors" Operator – A Vector Field Visualization Primitive. In *Proceedings IEEE Visualization 1999*, pp. 263–270. 1999. (Cited on page 10)

[PS08] R. Peikert, F. Sadlo. Height Ridge Computation and Filtering for Visualization. In I. Fujishiro, H. Li, K.-L. Ma, editors, *Proceedings of Pacific Vis 2008*, pp. 119–126. 2008. (Cited on pages 10 and 28)

[SCT+] A. R. Sanderson, G. Chen, X. Tricoche, D. Pugmire, S. Kruger, J. Breslau. Analysis of Recurrent Patterns in Toroidal Magnetic Fields. *IEEE Trans. Vis. Comput. Graph.*, (6):1431–1440. (Cited on page 10)

[SG89] A. Stettner, D. P. Greenberg. Computer graphics visualization for acoustic simulation. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '89, pp. 195–206. 1989. (Cited on page 10)

[SP07] F. Sadlo, R. Peikert. Efficient Visualization of Lagrangian Coherent Structures by Filtered AMR Ridge Extraction. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1456–1463, 2007. (Cited on pages 10 and 23)

[ST08] M. Sipos, B. G. Thompson. Electrodynamics on a grid: The finite-difference time-domain method applied to optics and cloaking. *American Journal of Physics*, 2008. (Cited on page 17)

[STS10] T. Schultz, H. Theisel, H.-P. Seidel. Crease Surfaces: From Theory to Extraction and Application to Diffusion Tensor MRI. *IEEE Transactions on Visualization and Computer Graphics*, 16(1):109–119, 2010. (Cited on page 10)

[SWTH07] J. Sahner, T. Weinkauf, N. Teuber, H.-C. Hege. Vortex and Strain Skeletons in Eulerian and Lagrangian Frames. *IEEE Trans. Vis. Comput. Graph.*, 13(5):980–990, 2007. doi:http://doi.ieeecomputersociety.org/10.1109/TVCG.2007.1053. (Cited on page 10)

[TS03] H. Theisel, H.-P. Seidel. Feature flow fields. In *Proceedings of Eurographics / IEEE TVCG Symposium on Visualisation*, pp. 141–148. 2003. (Cited on page 11)

[WSTH07] T. Weinkauf, J. Sahner, H. Theisel, H.-C. Hege. Cores of Swirling Particle Motion in Unsteady Flows. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1759 –1766, 2007. doi:10.1109/TVCG.2007.70545. (Cited on page 11)

[YST] T. Yokota, S. Sakamoto, H. Tachibana. Visualization of sound propagation and scattering in rooms. *Acoustical Science and Technology*, (1):40–46. (Cited on page 10)

**Declaration**

All the work contained within this thesis, except where otherwise acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

---

(Oliver Schmidtmer)