Hypersweeps, Convective Clouds and Reeb Spaces

Petar Hristov

School of Computing

University of Leeds

A thesis submitted for the degree of

Doctor of Philosophy

June 2022

This thesis is dedicated to my grandfather Petko Hristov. He taught me how to windsurf.

Тази дисертация е посветена на дядо ми Петко Христов. Той ме научи как да карам уиндсърф.

Acknowledgements

First and foremost, I would like to thank my supervisor Hamish Carr. He was not only a mentor, but also a friend. Hamish, thank you teaching me how to read and write and for supporting me all the way through.

I also want to thank my second supervisor Marc de Kamps for teaching me how to teach and for the many talks about the beauty of mathematics.

This thesis would not have been possible without my close collaborators - Daisuke Sakurai, Leif Denby, Gunther Weber and Oliver Rübel. You have all taught me how to collaborate and helped me become a better researcher. Gunther, thank you for inviting me to Berkley and mentoring me.

I would also like to thank my external assessor Ingrid Hotz and internal assessor Karim Djemame for the valuable feedback and fruitful discussions during my thesis defence.

My deepest gratitude to Lucy Godson. You have always believed in me and supported me one care package at a time. Thank you for all the memories and all the domats we shared.

To all the academics and PhD students and friends that were there for me - Jake Horsfield, Judith Clymo, Noleen Köhler, Samuel Wilson, Lukas Deutz and Ryan Ly. You have all made this journey worthwhile. It would not have been the same without all the board games, pints at the Library, the Ilkley school problem and talks about dissecting monkey brains.

I would like to thank my friends in the UK who have helped me so much along the way - Hristina Tsolova, Merlin Doherty, Robert Wakefield, Teodora Byanova, Martin Metodiev, Kaloyan Simeonov, Damyana Bojinova, Boyan Dimitrov, Ilko Ambarev and Diana Sofronieva. You have made Leeds feel like a home away from home.

Thank you to my friends in Bulgaria - Lyudmil Iliev, Atanas Angelov, Pavel Petrov, Rosen Stoyanov and Hristo Georgiev. You have made my trips back home feel truly special.

Finally, I'd like to thank my grandmother Atanaska Hristova, my father Georgi Hristov, my uncle Vladimir Hristov and my cousin Andrei Hristov. I did this in no small part for you and because of you.

Abstract

Isosurfaces are one of the most prominent tools in scientific data visualisation. An isosurface is a surface that defines the boundary of a feature of interest in space for a given threshold. This is integral in analysing data from the physical sciences which observe and simulate three or four dimensional phenomena. However it is time consuming and impractical to discover surfaces of interest by manually selecting different thresholds.

The systematic way to discover significant isosurfaces in data is with a topological data structure called the contour tree. The contour tree encodes the connectivity and shape of each isosurface at all possible thresholds. The first part of this work has been devoted to developing algorithms that use the contour tree to discover significant features in data using high performance computing systems. Those algorithms provided a clear speedup over previous methods and were used to visualise physical plasma simulations.

A major limitation of isosurfaces and contour trees is that they are only applicable when a single property is associated with data points. However scientific data sets often take multiple properties into account. A recent breakthrough generalised isosurfaces to fiber surfaces. Fiber surfaces define the boundary of a feature where the threshold is defined in terms of multiple parameters, instead of just one. In this work we used fiber surfaces together with isosurfaces and the contour tree to create a novel application that helps atmosphere scientists visualise convective cloud formation. Using this application, they were able to, for the first time, visualise the physical properties of certain structures that trigger cloud formation.

Contour trees can also be generalised to handle multiple parameters. The natural extension of the contour tree is called the Reeb space and it comes from the pure mathematical field of fiber topology. The Reeb space is not yet fully understood mathematically and algorithms for computing it have significant practical limitations. A key difficulty is that while the contour tree is a traditional one dimensional data structure made up of points and lines between them, the Reeb space is far more complex. The Reeb space is made up of two dimensional sheets, attached to each other in intricate ways. The last part of this work focuses on understanding the structure of Reeb spaces and the rules that are followed when sheets are combined. This theory builds towards developing robust combinatorial algorithms to compute and use Reeb spaces for practical data analysis.

Publications

Hristov, P., Weber G., Carr, H., Rübel, O. & Ahrens, J. (2020). Data Parallel Hypersweeps for In-Situ Computation. *Proceedings of the 10th Symposium on Large Data Analysis and Visualization (LDAV)*, 12-21. IEEE.

Hristov, P., & Carr, H. (2021). W-structures in contour trees. *Topological Methods in Data Analysis and Visualization VI* 3-18. Springer.

Contents

List of Figures

Part I

Introduction and Background

Chapter 1

Introduction

Scientific data obtained through measurements and computational simulations has to be analysed and visualised in order to be understood. Analysing data is a difficult task because data is massive, noisy, multiscale and multivariate. In order to tackle these challenges we need efficient algorithms which are based on rigorous mathematical theory. The output of these algorithms then needs to be visualised in a suitable format that facilitates the user's understanding.

Topology studies qualitative geometric properties such as the connectivity of areas of interest in data or features. By using algorithms from the fields of Topological Data Analysis or Computational Topology we can retrieve and visualise those features. Furthermore we can attribute a metric of significance to them in order to rank and discard all but the most important ones. This allows scientists to reduce the vast size of the data and focus their attention on what is significant to them.

Topological methods have found numerous applications in the field of scientific visualisation due to topology's close relation to geometry and analysis. The current state of the art has mostly focused on methods for analysing scalar, vector and tensor fields Heine *et al.* (2016), but few for multivariate fields.

Multivariate fields are an extension of scalar fields in the case where multiple scalar functions are defined on a common domain. The importance of multivariate fields has grown considerably in recent years with an increasing need to understand the structure and interactions of different properties of data. The goal of this thesis is to build towards the development of topological methods for multivariate topological data analysis.

Since multivariate topological methods build on scalar topological methods and are used to assist in multivariate visualisation, we must first be familiar with those. In this thesis we first contribute to scalar topological methods by scaling them to high performance computing systems. Then we contribute to multivariate visualisation methods by introducing a novel way of combining them with scalar visualisation methods and applying them to the study of convective cloud formation. Finally, we use what we have learned from those two contributions to build on the initial theory of a multivariate topological data structure called the Reeb space.

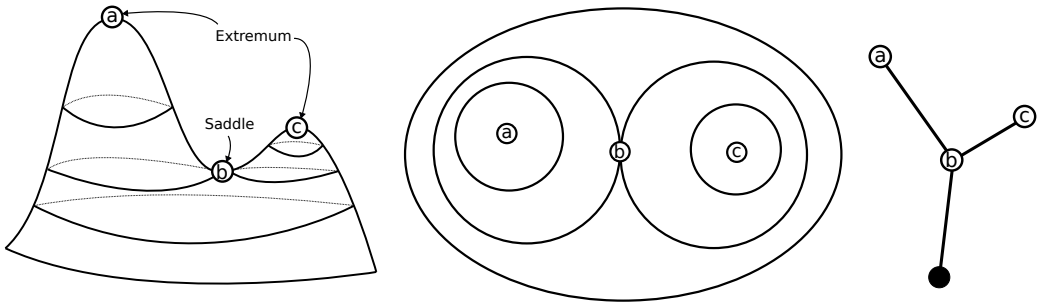
1.1 Isosurfaces and Contour Trees

The primary focus of this work is on scalar field and multi-field data. A scalar field attributes a single real value to each point in space. Mathematically, this is written as $f : \mathbb{R}^n \to \mathbb{R}$ where $\mathbb{R}^n$ is the domain and $\mathbb{R}$ is the range. A multi-field is a collection of scalar fields $(f_1, \ldots, f_m) : \mathbb{R}^n \to \mathbb{R}^m$ such that each component function f_i is scalar. Scalar fields and multi-fields are used for example in atmosphere science to model properties such as temperature, humidity and vertical velocity. They are used in numerical simulations to predict the weather and climate.

In two dimensions we can visualise the structure of a scalar field with its contour lines. Contour lines are curves in the plane where all the points on the curves have constant value. These are for example the lines we are familiar with on topographic maps. In a topographic map, the scalar field is the elevation of points in physical terrain.

We give an example of visualising the scalar field of elevation on a terrain in Figure 1.1. On the left we have shown the terrain in three dimensions by using an additional dimensions for the elevation. In the middle we have visualised the same terrain via some of its contours. Notice that the contours are concentric and never intersect. On the right we have the shown the relationships of the contours more abstractly with a graph called the contour tree.

The contour tree is a topological data structure that is obtained by shrinking all contours to single points. Thus all the contours on the left peak become points along the edge ab and all the contours on the right peak become points on the edge cb. The vertices on the contour tree are called critical points. This is where topological change happens as contours emerge (such as the peaks a and c) or where contours merge (such as the saddle b).



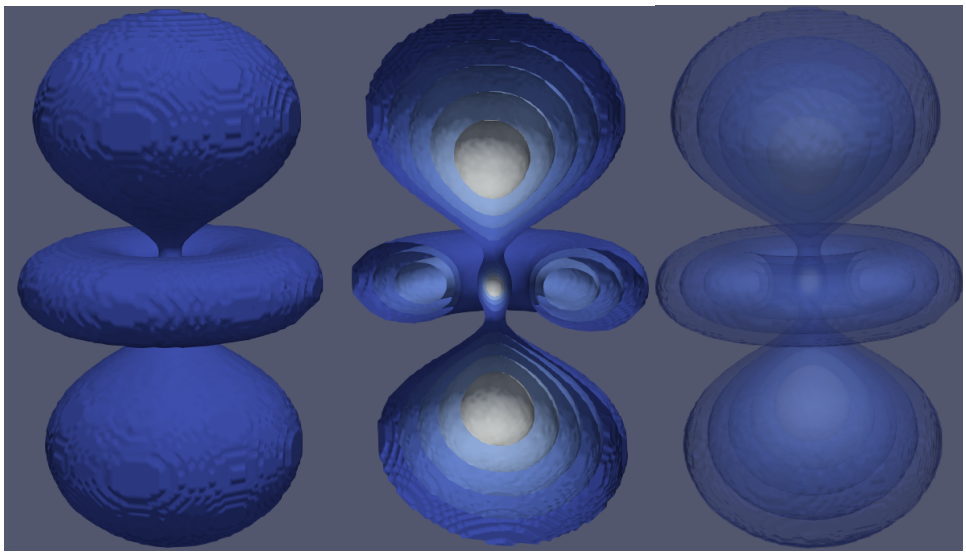
(a) Terrain in 3D.

(b) Terrain via contours.

(c) Terrain via contour tree.

Figure 1.1: The contour lines of a topographic map tell us how the terrain is structured. The contour tree is a compact representation of the topology of that structure.

In three dimensional domains the points of constant value are no longer curves, they are surfaces. These surfaces are called isosurfaces and they are extracted and rendered using computer graphics. You can see an example of this in Figure 1.2.



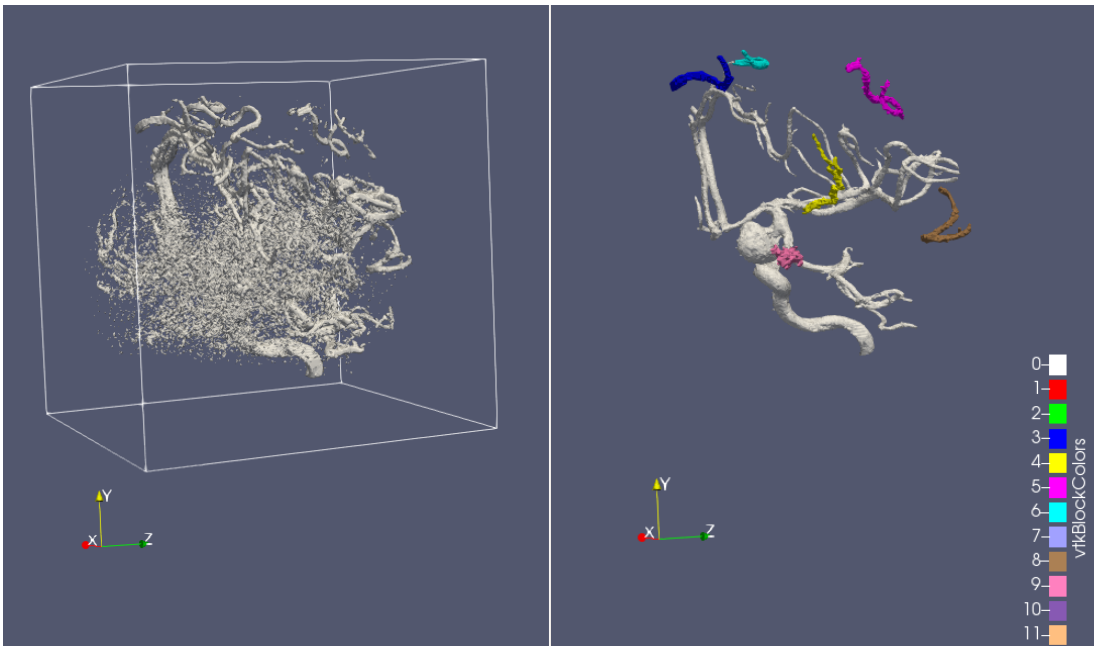
(a) Isosurface. (b) With Cutaway. (c) With Opacity.

Figure 1.2: The points of constant value in scalar field over a three dimensional domain are called isosurfaces. An isosurface is a surface in three dimensional space. Usually they are computed by a method from computer graphics called Marching cubes Lorensen & Cline (1987). These isosufaces come from a simulation of the spatial probability distribution of the electron in an hydrogen atom. Notice that to understand that field we need to plot multiple isosurfaces, but they are nested and occlude each other. To view all of them we can do a cutaway b) or reduce their opacity c).

Visualising and understanding the structure of a three dimensional scalar field is significantly more challenging. Unlike with contour lines we do not have the extra dimension in which to see all isosufaces at the same time. Furthermore, visualising multiple isosufaces is not always helpful, since they can be nested inside each other. We can deal with this to some extent by working with opacity or cutting through them, but that is time intensive and laborious (see Figure 1.2).

The contour tree provides a robust and systematic way to understand the structure of three or higher dimensional scalar fields because it always remains a one dimensional data structure. Since every line in the contour tree represents a connected contour in the data, we can attribute those with additional information. For example geometric information such as surface area, or volume or statistical information such as mean value or variance of a parameter. Then we can automatically extract the contours with user chosen properties and visualise them with a process called contour tree simplification.

1.2 Hypersweeps for Contour Tree Simplification



(a) Noisy isosurface. (b) Most significant contours.

Figure 1.3: Isosurfaces from real world data sets often contain noise and sampling error which prevents coherent visualization. Our method removes that and automatically identifies significant features. This data set is an x-ray scan of the arteries of the right half of a human head featuring an aneurysm which can be identified with our method.

Contour tree computation and simplification is well established in the literature, however current algorithms for simplification are not scalable. By not scalable we mean that they are not designed to utilise multiple cores in modern day high performance computing systems such as supercomputing clusters. The scalability of these contour tree simplification algorithms is crucial because otherwise we cannot apply them to the large data sets produced by these supercomputers.

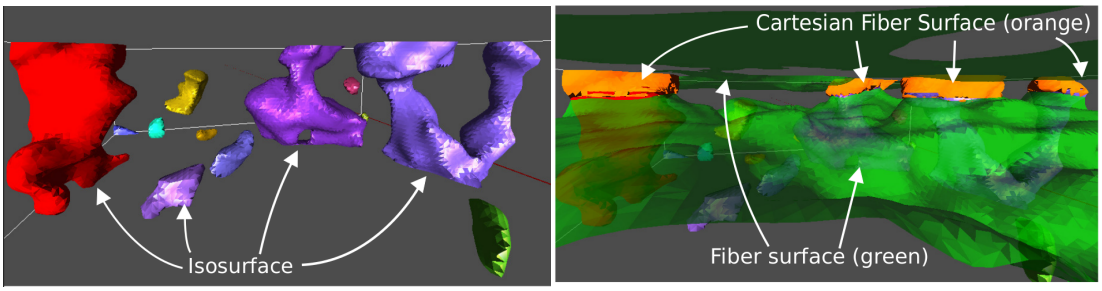
Our first contribution is to introduce parallel algorithms for simplifying the contour tree to discover significant features in scalar field data. We developed a method we call a hypersweep to compute the geometric and statistical measures over the contour tree. Then we replaced the serial method of using those measures to simplify the contour tree called branch decomposition. We implemented those algorithms in the open source VTK-m library for public use.

Using our open source implementation, we developed the first fully parallel pipeline that uses contour tree analysis. The first part of the pipeline runs on the computing cluster along with the numerical simulations. We compute the contour tree, simplify it

and extract the most significant contours. In order to save bandwidth we store images of those contours for visualisation purposes with a database called Cinema.

In the second part of the pipeline the user downloads the images and reconstructs the original surfaces. This can be done on commodity hardware to enable use of access. We evaluated our application on various real world data sets, you can see on example in Figure 1.3.

1.3 Trivariate Visualisation of Convective Cloud Formation



(a) Isosurface (b) Cartesian Fiber surface

Figure 1.4: Given an isosurface a) and a Fiber surfaces (b) in green, the Cartesian fiber surface is the boundary of the volume contained in the two. The Cartesian fiber surface is used to refine features defined with one method (isosurfaces) by subfeatures defined in another method (Fiber surfaces).

In real world data there is an increasing need to understand the structure of multi-fields as opposed to the structure of multiple individual scalar fields. For example the process of convective cloud formation is governed by multiple physical variables such as temperature, humidity and vertical velocity. This process cannot be described entirely by any one of those individual variables, but only by their join contribution to the phenomenon. The need to analyse multi-field data has led to the generalisation of isosurfaces to bivariate (two scalar fields) fields via fiber surfaces and to general multifields via feature level sets.

For bivariate fields the range is no longer the real line like for isosurfaces, but a two dimensional plane. Hence we define fiber surface with a polygon, not a single value. Then the fiber surface is the surface in the domain that separates the points which take values inside the polygon from the ones that take values outside. Feature level sets generalise this to any multi-field, and we can define any high dimensional piece of geometry in the range, not just a polygon.

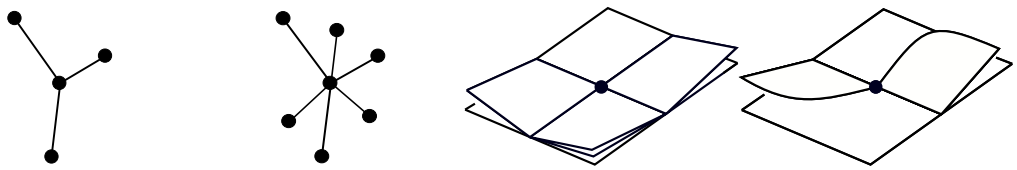
As the dimension of the range increases, it is difficult to define high dimensional

geometry that captures the areas of interest in data. Furthermore there isn't an efficient way to combine multiple areas of interest, defined by the various methods of isosurfaces, fiber surfaces and feature level sets. Our second contribution is to develop a solution to both of these issues with Cartesian fiber surfaces and apply that to analysing convective cloud formation.

Using Cartesian fiber surfaces we collaborated with scientists from the University of Leeds to develop an application for analysing the process of convective cloud formation. Convective clouds are significant in predicting the weather and climate, but their formation is not yet fully understood. The difficulty lies in determining the exact relationship based on interactions of temperature, vertical velocity and moisture that produces clouds.

By combining isosurfaces and fiber surfaces via Cartesian fiber surfaces we helped atmosphere scientists visualise two novel findings. The first one is about areas in the environment that encourage and inhibit cloud formation and the second is about the internal structure of features that trigger convective cloud formation. You can see an example of using the application in Figure 1.4.

1.4 Generating Reeb Spaces Neighborhoods



(a) Contour tree. (b) Contour tree with more nodes. (c) Reeb space. (d) Reeb space with a different sheet arrangement.

Figure 1.5: An example of the increase in complexity from contour trees to Reeb spaces. In contour tree a) the middle vertex has two neighbours above it and one below it. In general when describing the neighbourhoods we can specify the number above and below. For example in contour tree b) the middle vertex has three above and three below. The structure of a Reeb space is more complex as you can see in in c) and d). In Reeb spaces we have quadrants each with a different number of sheets. Furthermore, more the sheets can attach in different ways and appear, disappear and merge in different configurations.

As we have seen so far, isosurfaces and their accompanying data structure the Contour tree are well established as a key tool in scientific visualisation for understanding the structure of a scalar field. We saw that we can extend isosurfaces to bivariate fields via fiber surfaces and to general multifields via feature level sets. However, the

accompanying data structure that would enable better fiber surfaces selection is not as well understood as the Reeb graph. This data structure is called the Reeb space.

The Reeb space is a polyhedron made up of vertices, edges connecting the vertices and two dimensional sheets, which connect the edges. In Figure 1.5 we have shown why the structure of the Reeb space is more difficult to describe that of the contour tree. Algorithms exist for computing low-dimensional Reeb spaces, but a complete analysis of their structure for practical data has to date been absent. We propose a fully combinatorial method of computing all possible local structures in a Reeb space, parametrised by the edge and vertex degree of the input mesh. Unlike previous methods, our method makes no assumptions about the input mesh aside from simple conditions on the data values on the vertices, which can be guaranteed by standard perturbation techniques such as simulation of simplicity. We demonstrate our method by giving a full classification of all Reeb spaces neighbourhoods where the input mesh has maximum vertex degree six.

1.5 Overview

This thesis is divided into several parts. Part I introduces the research topics and major contributions, reviews the mathematical background and provides a review of the relevant academic literature. Part II introduces the first contribution of this work, which is data-parallel contour tree operations via hypersweeps. Part III covers our second contribution, that of combining isosurfaces and fiber surfaces via Cartesian fiber surfaces for visualising convective cloud formation. Party IV describes our third contribution - algorithms for generating and classifying Reeb space neighbourhoods of PL maps. Finally, Part V gives our conclusions and future work.

Chapter 2

Background

In this chapter we will present the relevant mathematical theory for understanding the rest of the thesis. We assume that the reader has a good understanding of multivariate calculus and linear algebra. Otherwise we refer the reader to Spivak (2018) for a modern and concise introduction to both subjects. Beyond this we will introduce any notions we use from topology, geometry, and abstract algebra.

2.1 Point Set Topology

Topology is the framework on top of which mathematicians build continuous spaces and functions that are used in analysis and linear algebra. The topology of a set defines a structure that tell us when two elements are close or near to one another. The structure defined by topology is given in terms of open sets - elements which are close to one another are part of an open set.

Definition 2.1. *Let X be a set and τ be a set of subsets of X. The set τ is a topology on X when the following holds:*

- X and $\emptyset \in \tau$.
- If U and $V \in \tau$ then $U \cap V \in \tau$.
- If $\{U_\lambda\}_{\lambda \in \Lambda}$ is a family of subsets of X, where $U_\lambda \in \tau$ for all $\lambda \in \Lambda$, then $\bigcup_{\lambda \in \Lambda} U_\lambda \in \tau$.

The elements of X are called points and the elements of τ are called open sets, or simply open. The complement of an open set is called a closed set, or simply closed. The topology of a set X is by no means unique. For example the topology of X can consist of all subsets of X or just X and the empty set.

The standard topology in Euclidean space $\mathbb{R}^n$ is given in terms of open balls. Given a point $x = (x_1, x_2, \ldots, x_n)$ in $\mathbb{R}^n$ the open ball around x of radius ϵ is defined as

$B_\epsilon(x) = \{y \in \mathbb{R}^n : d(x,y) < \epsilon\}$ where d is the standard Euclidean distance $d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$. Then the topology of $\mathbb{R}^n$ is then given by all the open balls around all points will all possible radii, as well as their unions and finite intersections.

The open sets in a topology however can be difficult to describe. This is why it's useful to describe them with a smaller subset of the open sets. This is called a base of the topology, and is analogous to a base for a vector space in linear algebra.

Definition 2.2. *A base for a topology τ is a collection of subsets $\beta \subset \tau$ such that every element in τ s equal to a union of elements in β.*

A topological space that has a countable base is said to be second countable. This is useful property since many of the more well understood topological spaces such as Euclidean space are second countable. Another useful property is that a topological space is Hausdorff.

Definition 2.3. *A topological X space is Hausdorff if for any two distinct elements $x, y \in X$ there exist neighbourhoods $x \in U$ and $y \in V$ such that $U \cap V = \emptyset$.*

Euclidean space is also Hausdorff since for any two distinct points we can pick two balls around them whose radius is smaller than the distance between the points divided by two.

Next we will define the class of functions that preserve topological properties of spaces. Those are continuous functions.

Definition 2.4. *A function $f : X \to Y$ is said to be continuous when the preimage of an open set in Y is an open set in X.*

When we apply this definition to the topology of Euclidean space we obtain the standard definition of continuity we have from calculus. Furthermore if f is a bijection and its inverse is continuous then f is called a homeomorphism. Homeomorphisms play a central role in topology because when there exists a homeomorphisms between two spaces, that means they have the same topological properties. Those spaces are called homeomorphic and topologists as usually interested in studying topological spaces up to homeomorphisms.

A key idea in calculus and analysis is that of limits and convergence. Limits and convergence can be defined more generally for topological spaces as follows:

Definition 2.5. *Let X be a topological space, let $x \in X$ and let $\{x_n\}_{n\in\mathbb{N}}$ be a sequence of points in X. The sequence converges to x and we call x the limit of the sequence when for every neighbourhood U of x there exists $N \in \mathbb{N}$ such that $x_n \in U$ for all $n \geq N$.*

The idea behind this definition is that all points in the sequence become closer to the limit. In a Euclidean space we substitute the neighbourhoods with open balls to obtain the usual definition from analysis. An important use of limits is to approximate the elements of a set via the elements of one of its subsets.

Definition 2.6. *Given a set X and a subset of X called Y, the closure of Y in X consists of the points in Y as well as all the points in X that all sequence in Y converge to. The set Y is said to be dense in X.*

Dense subsets play an important role in mathematics because you can use them to study complicated objects via a more well understood subset of those objects. For example real numbers can be defined using an equivalence class on the converging sequence of rational numbers using Cauchy sequences. Another example is studying the class of all differentiable functions by a restricted class of differentiable functions which are well understood called Morse functions (see 2.2).

A topological invariant is a property of a topological space that is preserved under continuous functions. One such topological property is path-connectedness. A path-connected topological space is one where any two points can be connected with a path.

Definition 2.7. *Let X be a topological space and let $x, y \in X$ be any two points. A path between x and y in X is a continuous function $f : [0, 1] \to X$ such that $f(0) = x$ and $f(1) = y$.*

Definition 2.8. *A topological space X is said to be path-connected if there exists a path between any two points $x, y \in X$.*

Another important topological invariant is compactness. For a topological space X and a subset $A \subset X$, we define an open cover of A as the indexed family $\{U_\alpha\}$ for which $A \subset \cup\{U_\alpha\}$.

Definition 2.9. *A topological space X is called compact when every open cover of an any subset of X has a finite subcover.*

Compactness is important because it captures the idea of infinite sets which are not "too big". For example, in Euclidean space any compact set is closed and bounded. That means that those sets have a boundary and they can be contained in an open ball with finite raidus.

One way to obtain new topological spaces is through quotient spaces, which are defined via an equivalence relation. An equivalence relation $\sim$ on a topological space X partitions the points of X into equivalence classes. The equivalence class of a point $x \in X$ is defined as the set $[x] = \{y \in X : x \sim y\}$. The set of equivalence classes is the quotient of X by $\sim$ and is written as as $X/\sim$.

Definition 2.10. *Let X be a topological space and $\sim$ be an equivalence relation defined on X. The quotient topology of $X/\sim$ is formed by the sets $U \subseteq X/\sim$ such that $\pi^{-1}(U)$ is open in X.*

For an example of how a quotient space changes the topology of a space consider the closed disk $D = \{x^2+y^2 \leq 1\}$ and its subset the circle $S^1 = \{x^2+y^2 = 1\}$. The quotient

space D/S^1 is homeomorphic to the three dimensional sphere $S^2 = \{x^2 + y^2 + z^2 = 1\}$. This is because D/S^1 identifies all points on S^1 as a single point and glues them together. This closes the boundary of D and turns it into a sphere.

One of the most important types of topological spaces are manifolds - a mathematical generalisation of curves and surfaces.

Definition 2.11. *A topological space X is a manifold of dimensions n (or a d-manifold) if:*

- *Every point in X has an neighbourhood that is homeomorphic to $\mathbb{R}^d$.*
- *X is Second countable.*
- *X is Hausdorff.*

Zero dimensional manifolds are points; one dimensional manifolds are lines, circles and curves; two dimensional manifolds are the surfaces we are familiar from geometry such as the sphere and the torus.

A manifold with a boundary is a manifold with both interior and exterior points. The interior points are homeomorphic to $\mathbb{R}^n$ and the exterior points (the ones on the boundary) are homeomorphic to half of $\mathbb{R}^n$ which we can express as $\{(x_1, \ldots, x_n) | x_i \geq 0\}$. The boundary of a manifold M is written as ∂M and is itself an $n-1$ dimensional manifold. A compact manifold without a boundary is called closed.

An example of a manifold with a boundary is the closed interval $[0, 1]$ where the boundary is the endpoints $\{0, 1\}$. Another example is the closed ball $B_\epsilon(x) = \{y \in \mathbb{R}^3 : d(x, y) \leq \epsilon\}$ whose boundary is the 2-sphere.

It is often difficult to analyse the topology of a space by just considering its open sets. This is why in the following two sections we will employ additional tools from other fields of mathematics to aid in our analysis of the topology of a space. These tools are differentiable functions over differentiable spaces and combinatorial approximations of topological spaces.

2.2 Morse Theory

Morse theory is the study of differentiable manifolds via differentiable maps defined on them Matsumoto (2002a); Milnor (2016). A core idea in Morse theory is to study the critical values of a scalar map $f : M \to \mathbb{R}$ where M is a differentiable manifold. However this is very complex to do in general for smooth maps because some critical points can be degenerate and have complicated behaviour. Therefore we restrict our attention to a special class of differentiable functions called Morse functions.

Definition 2.12. *A function $f : M \to \mathbb{R}$ is a Morse Function if f is smooth and at critical points the Hessian (matrix of second partial derivatives) is full rank.*

The points in M where the first derivative is equal to zero are called critical points, other points are called regular. We use the critical points of M to decompose it into a family of path-connected submanifolds with equivalent topology. These are the level sets, sublevel sets and superlevel sets.

Definition 2.13. *A level set at a value h of a Morse function $f : M \to \mathbb{R}$ is the set $f^{-1}(\{h\}) = \{x \in M : f(x) = h\}$*

Sublevel sets are are defined as the preimage of an interval instead of a single value $f^{-1}((-\infty, a]) = \{x \in M : f(x) \in (-\infty, a]\}$ and superlevel sets are defined analogously as preimages of intervals of the form $[a, \infty)$.

Morse functions ensure the following properties hold:

- None of the critical points are degenerate.
- Changes in the topology of level sets, sublevel sets and superlevel sets only happen at critical points.
- A Morse function defined on a surface has a finite number of critical points.

Morse functions allow us to decompose a manifold into its level sets. We will use the theory we have developed so far to introduce the principal tool that allows us to analyse how the connectivity of level sets $f^{-1}(\{h\})$ changes as we vary the input parameter h.

The Reeb graph is a tool that encapsulates the evolution of the topology of level sets of a continuous function. When the function is Morse, an edge in the Reeb graph corresponds to a sequence of contours in the level sets whose topology does not change. The vertices correspond to critical points where the topology of those components does changes. An example of a topological change is when connected components in the level sets appear or dissapear or when two connected components split or merge. Morse theory ensures that critical points occur at distinct values of the parameter and are isolated. This removes ambiguities that may arise in the construction of the Reeb graph.

Definition 2.14. *Given a topological space X and a continuous function $f : X \to \mathbb{R}$ we can define an equivalence relation $\sim$ such that two points x, y in X are equivalent when there exists a path between them in a level set $f^{-1}(\{h\})$ for some $h \in \mathbb{R}$. The Reeb graph is the quotient space $X/\sim$ together with the quotient topology.*

We can think of the Reeb graph of a space X as the quotient space where the connected components of all level sets are contracted to a single point. The resulting topological graph can also be though of as a discrete graph. To do so, we must enumerate the vertices and record all edges between them.

In the next section we will take a look at certain tools from Algebraic Topology that allows us to translate the continuous mathematical results we have obtained so far into the realm of finite combinatorial structures that would allow us to perform actual computation.

2.3 Fiber Topology

Studying more than one scalar function at the same time, instead of individually with Morse theory, becomes more complex. This is the subject of study of Fiber Topology. For a comprehensive introduction to the subject we refer the reader to Saeki (2004) and for a more concise introduction we refer the reader to Saeki (2017). In this section we will outline some of the main definitions and results.

Let N be a smooth manifold and let $f : N \to \mathbb{R}^n$ be a multi-field. By a multi-field we mean a set of scalar maps $f_i : N \to \mathbb{R}$ for $i \in \{1, \ldots, n\}$ such that $f = (f_1, \ldots, f_n)$. A smooth map is called stable when there exists a neighbourhood of the map in the space of smooth maps such that the map is equivalent to all those maps up to diffeomorphism. That means that if the map is perturbed slightly the resulting map behaves the same up to diffeomorphism on the domain and the range. Furthermore for the case of a bivariate field (f, g) the space of smooth maps is dense in the space of stable maps (Recall Definition 2.1). That means that any smooth map can be approximated arbitrarily well by a stable map.

The differential of a multi-field (f, g) is the linear map associated with the Jacobian matrix of (f, g) at a particular point p. If the rank of the Jacobian is less than the dimension of the domain manifold or of the range, then the point is called singular. This is analogous to critical points for Morse functions. The structure of the set of singular points for a bivariate field consists of a set of curves in the domain called folds, connected by points called cusps.

The preimage of a single point in the range is called a fiber, and it is denoted as $(f, g)^{-1}(p)$ where $p \in \mathbb{R}^2$. When p is a regular point we call the fiber regular, and when p is a singular point we call the fiber singular. This is analogous to the definition of level set in Morse Theory. In Fiber Topology two fibers are considered equivalent when we can establish a diffeomorphism on neighbourhoods of the fibers, not just the fibers themselves. Fiber topology is concerned with classifying fibers up to equivalence, which is why the term fiber itself carries more information than just the preimage.

One key result that describes the behaviour of fibers is the Ehresmann Fibration Theorem. A corollary of that theorem is that the equivalence type of a fiber only change at the singular set of the bivariate field. Suppose we split the range into singular regions, the image of the singular set and regular regions, all other points. The singular regions will be a set of curves that overlap and regular regions will be two dimensional regions segmented by the singular regions. All fibers in a regular region are equivalent and they change their equivalence type as they pass through a singular region.

The generalisation of the Reeb graph for a multi-field is called the Reeb space. The Reeb space is the quotient space (recall Section 2.1) obtained by contracting the connected components of all fibers to a single point. The Reeb space is also called the Stein factorizaion and it has played an important role in the global study of stable maps. The Reeb space can be obtained from a classification of singular fibers by considering the connected components of the regular fibers in their equivalence region.

The Reeb space is no longer one dimensional like the Reeb graph. For example for

a bivariate function the Reeb space is a two-dimensional complex. That means that it is comprised of a number of surfaces glued together by singular curves. More generally the Reeb space is stratification of manifolds, but it has been shown for stable maps the Reeb space is a polyhedron.

A polyhedron can be represented as a combinatorial structure called a triangulation. This is essential for the purposes of computation, which is the ultimate goal of this work. In our next section we move from away smooth mathematics to discrete mathematics.

2.4 Piecewise Linear Topology

In the previous sections of this chapter we presented the smooth mathematical theory that underlies our investigation. However, in order to compute anything in practise we are going to need a combinatorial representation. For a comprehensive introduction we refer the reader to Rourke & Sanderson (1972) and for a more concise modern introduction we refer the reader to Edelsbrunner & Harer (2022). Here we will outline the main definitions and results.

Simplical Complexes are one of the first combinatorially flavoured topological spaces one encounters in Piecewise Linear Topology. A simplicial complex is a subset of $\mathbb{R}^n$ that consists of points, line segments, triangles and their higher dimensional analogues attached to one another in a single geometric object. In order to understand simplicial complexes we must first define their basic building blocks.

Definition 2.15. *Let $\{v_1, \ldots, v_k\}$ be k points in $\mathbb{R}^n$. A convex combination of the points is a sum $\sum_{i=1}^{k} \lambda_i v_i$ where $\lambda_i \geq 1$ and $\sum_{i=1}^{n} \lambda_i = 1$.*

If we decide to take the subset of $\mathbb{R}^n$ covered by all possible convex combinations we obtain the convex hull of the points. Another way to construct simplices is to inductively build them up of simplices of lower dimesion using the cone operation. The cone operation takes two sets in Euclidean space and all line segments between them.

Definition 2.1 (Cone)**.** *Given two sets A and B in Euclidean space, the cone$(AB) = \{ta + (1-t)b \mid t \in [0,1], a \in A$ and $b \in B\}$ consists of all line segments between every point in A and every point in B including the endpoints of the line segments endpoints.*

Definition 2.16. *Let $\{v_1, \ldots, v_k\}$ be points in $\mathbb{R}^n$ in general position. The set of all convex combination of those points is called the $k-$simplex defined by the points. We will write that simplex as $[v_1, \ldots, v_k] \subset \mathbb{R}^n$*

The coefficients in the convex combination are called barycentric coordinates. The number k is also called the dimension of the simplex. We will call the simplices of dimension $0, 1, 2$ and 3 vertices, edges, triangles and tetrahedron respectively.

A face of a simplex is the convex hull of a non-empty subsets of its points. For example the faces of the tetrahedron are the four triangles, six edges and four vertices.

To construct a simplical complex all we have to do is take the union of a number of simplices and "glue" them together along common faces without allowing self-intersection.

Definition 2.17. *A simplical complex K is a finite collection of simplices, such that if τ is a simplex K then all faces of τ must be simplicies in K. Furthermore the intersection of two simplicies in K is either empty or a common face of both.*

Most of the data sets we will consider in this work will be simplicial complexes. In a simplicial complex we need to define the neighbourhoods of points, these are analogous to the open sets in the smooth case.

Given a simplicial complex K we use $|K|$ to denote the underlying geometric space which is defined as $|K| = \cup_{\sigma \in K}(\sigma)$ equipped with the subspace topology (Recall 2.1). A subcomplex M of K is a simplicial complex such that every simplex of M is a simplex of K. The k-skeleton of a simplicial complex is the subcomplex that includes all simplices of dimensions k or lower. We write the k-skeleton of a simplicial complex K as K^n. In this notation K^0 is the set of vertices, K^1 is the set of edges and so on.

Definition 2.2 (Star, Link, Closed Star)**.** *The star of a simplex σ consists of all simplices which have σ as a face. In general the star is not a simplicial complex, so we also define the closed star to include the faces of the simplices in the star. The link of an simplex consists of the simplices in the closed star which have an empty intersection with the simplex σ.*

We will not differentiate between the terms star and the closed star, we assume that the star contains all its faces. We say that a simplicial complex triangulates its underlying geometric space. The analogous topological spaces to manifolds in PL topology are combinatorial manifolds.

Definition 2.3 (Combinatorial Manifold)**.** *A combinatorial d-manifold is a triangulation of a d-manifold such that the link of very vertex triangulates the (d-1)-sphere and is itself a combinatorial (d-1)-manifold.*

Another more general type of PL space is a polyhedron.

Definition 2.4 (Polyhedron)**.** *A subset of $\mathbb{R}^n$ is a polyhedron P when every point $p \in P$ has a cone neighbourhood $N = cone(p, L)$ where $L \subset P$ and L is compact.*

Analogously to simplicial complexes N is called the star of p and L is called the link. All simplicial complexes are polyhedrons and all polyhedrons are triangulable.

In differential topology the natural maps between spaces were infinitely differentiable, or smooth maps. In PL topology the natural maps are PL maps.

Definition 2.5. *A map $f : K \to \mathbb{R}^n$ where K is a simplicial complex is a PL map if each point $a \in K$ has a star $N = cone(a, L)$ such that $f(\alpha a + \beta x) = \alpha f(a) + \beta f(x)$ where $x \in L$ and α and β are a convex combination.*

In a sense PL maps are locally conical, straight lines from the local cone structure are mapped linearly. For simplicial complexes it is enough to define the values of a PL map on the vertices, or on the 0-skeleton. The value for all other points on all other simplices are taken via their barycentric coordinates.

Given a function defined on the vertices of the complex $f : K^0 \to \mathbb{R}$ we can define a PL function on the whole of the simplicial complex $h : K \to \mathbb{R}$ as follows. Any point in any simplex can be written as a convex combination of the vertices of that simplex. For example for a point in a simplex $x = \sum_{i=1}^{i=n} \beta_i v_i$ spanned by $[v_1, \ldots, v_n]$ we define the PL function on the simplicial complex as $h(x) = \sum \beta_i f(v_i)$. This satisties the PL definition of a function 2.4.

When a PL map is defined on simplicial complex we can also define the lower an upper link of a vertex. The upper link consists of all simplices in the link whose spanning vertices have a larger function value than the vertex. Analogously the lower link is defined as all simplices in the link whose spanning vertices have a lower function value than the vertex.

Chapter 3

Literature Review

In this chapter we give an overview on the topic of topological data analysis in Scientific visualisation. The aim is to broadly cover the relevant theory, concepts, and methods that have been developed in the field so far. We present algorithms, where they are available, as well as their asymptotic analysis and practical implementations. For each method we also outline the applications it has found in analysing and visualising real world data.

3.1 Scalar Field Methods

In this section we will introduce one of the primary visualisation techniques for scalar fields called an isosurface. Formally, a scalar field is a real valued continuous function $f : \mathbb{R}^n \to \mathbb{R}$. It assigns a real number to every point in n dimensional Eucledian space. A level set of a scalar field f at an isovalue $c \in \mathbb{R}$ is the set of points in $\mathbb{R}^n$ that are mapped to the same constant value c. Analogously we define the sublevel set as the set a points which are mapped to a value less than c and the superlevel set as the set points mapped to values bigger than c. An isosurface is then defined as a level sets of a three dimensional domain.

In practise we cannot work with the entire domain and scalar function because they are continuous. Instead, we use a discrete approximation, built on top of a finite set of sampled points. If we assume that the points are sampled at the vertices of a regular grid we can interpolate to "fill" the missing function values at the edges, faces and interiors of the grid. If the grid is made of tetrahedra the standard method is barycentric interpolation and if it is made up of cubes we often use trilinear interpolation. Computational considerations for irregular grids are outside the scope of this work.

Once we have produced an approximation of the underlying real valued function we note that a level set $f^{-1}(c)$ separates the domain into two disjoint sets Wenger (2013a). They consists of the points that take a higher value and the points that take a lower value. This property enables one of the first and most widely used algorithms for isosurface extraction called Marching Cubes Lorensen & Cline (1987). Marching Cubes

works by dividing the domain into cubes and constructing a triangulated approximation of the intersection of the isosurface with each cube independently. We inspect the vertices of every cube to determine whether their value is bigger or smaller than the isovalue. If one vertex of an edge has a bigger isovalue and the other one smaller, then we know that the isosurface intersects that edge. If we assume that every vertex is either bigger or smaller there are $2^8 = 256$ possible configurations. Each configuration corresponds to a specific isosurface construction pattern within that cube. The number of distinct configurations can be reduced down to 24 by considering rotational, reflective and mirror symmetries van Gelder & Wilhelms (1994). Finally, in order to construct the isosurface patch within the cube we compute the point of intersection along the edge via linear interpolation.

The original Marching Cubes paper Lorensen & Cline (1987) is one of the most cited papers in computer graphics and visualisation and there have been numerous extensions to improving its computational efficiency, extending the output and input and ensuring correctness and consistency of the geometry and topology of the output Newman & Yi (2006). One of the most relevant extension is Marching Tetrahedra Shirley & Tuchman (1990). Marching Tetrahedra subdivides each cube in the rectilinear grid into multiple tetrahedra. As a tetrahedron has only four vertices, there are $2^4 = 16$ possible configurations which are reduced down to 3 when we consider reflection and rotation Treece *et al.* (1999a). The tradeoff of having less cases and simpler geometrical objects is that the choice of subdivision scheme produces geometric artifacts Carr *et al.* (2001) and increases computation time by a factor of at least five or more (depending on how many tets are used for each cube). In Section 3.7 we will present an extension of isosurfaces to the bivariate case called Fiber surfaces and show how a variant of Marching Tetrahedra is used to compute them accurately and efficiently.

3.2 Scalar Field Topology

The core idea of Topological Data Analysis is to use topological structures to define and extract features of interest in data. In this section we will consider the three major topological structures that have found the widest use in scientific visualisation. Those are Persistent homology, the Morse-Smale complex and the Reeb graph.

Persistent homology Edelsbrunner & Harer (2008) captures the evolution of the Homology Groups of a filtration of a simplicial complex. Homology groups are studied in Algebraic Topology as a means of describing the connectivity of a topological space. Elements of the homology groups represent the connected components, holes, voids and their high dimensional equivalents. Within this framework they are referred to as 0-cycles, 1-cycles, 2-cycles, and n-cycles respectively. A filtration of a simplicial complex is a sequence of complexes that start from the empty set and iteratively build up to the final complex by attaching simplicies at every step. For example when we attach new simplices new connected components may appear and when two connected components are bridged together they become one and one of them disappears. The same goes for the higher dimensional cycles - they appear at certain points in the

filtration, they persist for a certain number of steps and if they are filled in at any point they disappear. An element that persists for a large number of steps in the filtration is called persistent and deemed significant, while an ephemeral element is considered noise and can be discarded. Persistent homology is computed with a procedure that amounts to Gaussian elimination on the boundary matrix of the filtration. Using the boundary matrix, one sums up simplices to track the birth, persistence and death of cycles as they are formed and subsequently filled. The computational complexity of Persistent Homology is shown to be equivalent to matrix rank computation Edelsbrunner & Parsa (2014). Persistent homology has found various applications in neuroscience Reimann *et al.* (2017), computational biology Verovšek & Mashaghi (2016) and astrophysics J. Adler *et al.* (2017).

The Morse-Smale Complex is a topological data structure that is made by considering integral lines, or maximal curves along a manifold whose tangent vectors match the gradient. In Morse Theory these curves are obtained as the solution to a first order differential equation using the gradient field of the manifold. From the theory of differential equations we know that these integral lines exist and that they do not intersect, and that their origin and destination are critical points. The Morse-Smale complex is a partition of the manifold into regions where the integral lines share a common origin and destination. Algorithms that compute the Morse-Smale complex rely either on piecewise-linear Morse Theory or the combinatorial reformulation of Morse Theory called Discrete Morse Theory Scoville (2019). Methods based on the piecewise-linear case obtain the cells of the Morse-Smale complex by tracing out paths of steepest ascent and descent along the edges of the input mesh Edelsbrunner *et al.* (2003a,b) These algorithms are computationally intensive and not well parallelisable. An improvement on this are algorithms based on Discrete Morse theory. They compute the cells of the Morse-Smale complex explicitly either by growing maximal regions from seeds in cells containing minima and maxima Gyulassy *et al.* (2007) or by a divide and conquer approach that computes the discrete gradient vector field for parts of the data and combines them appropriately Gyulassy *et al.* (2008). Such algorithms achieve an average case running time of $O(nlogn)$. The Morse-Smale complex has been applied to molecular shape analysis Cazals *et al.* (2003), automated segmentation in histopathology Robins *et al.* (2011) and terrain modeling Danovaro *et al.* (2007).

The Reeb graph Reeb (1946) represents the topological skeleton of an object. It was initially introduced in the field of Differential Topology Milnor (2016) to study manifolds via real valued functions defined on them. The Reeb graph is the quotient space obtained by contracting the connected components of the level sets of the function to a single point. The nodes of the Reeb graph correspond to critical points in the domain where topological events occur, while the edges correspond to features which are separated by these topological events. Examples of topological events are local minima and maxima where connected components of level sets appear and disappear and saddle points where connected components of level sets merge together or split apart. The Reeb graph was introduced in the field of Computer Graphics in 1991 as a discrete data structure for surface coding Shinagawa *et al.* (1991). Since then it has found numerous

applications in the fields of computer graphics, scientific visualisation and computer vision. Those include shape matching and encoding Attene *et al.* (2006); Hilaga *et al.* (2001a), compression Biasotti *et al.* (2000), surface segmentation Sorgente *et al.* (2018) and parametrisation Steiner & Fischer (2001); Zhang *et al.* (2005), isosurface remeshing Wood *et al.* (2000) and simplification Wood *et al.* (2004), feature extraction Bajaj *et al.* (1997), transfer functions for volume rendering Weber *et al.* (2007b) and topological simplification Weber *et al.* (2007b); Wood *et al.* (2004). A survey on its further application is Biasotti *et al.* (2000).

From these three approaches only the Reeb graph has been successfully generalised to higher dimensions. The output of one dimensional Persistent Homology is complete in the mathematical sense in that it gives a full description all topological information within a filtration. A similar result is not possible for a filtration described by multiple parameters, or a multifiltration Carlsson & Zomorodian (2009). This negative result is partially offset by the definition of an two alternative invariants - the rank invariant Carlsson *et al.* (2009b) and persistent Betti numbers Cerri & Landi (2013). However, they are not complete, and only approximate the Betti numbers of the multidimensional filtration. This is further complicated by the fact that computing multidimensional persistence has a worst case running time of $O(n^4m^3)$ Carlsson *et al.* (2009b). To our knowledge there is no extension of the Morse-Smale complex to higher dimensions. The Reeb graph, on the other hand, has been generalised to the Reeb space, which we will descibe in detail in Section 3.9.

3.3 Reeb Graph algorithms

The first algorithm for Reeb graph computation on 2-manifolds is due to Shinagawa and Kunii Shinagawa & Kunii (1991). The algorithm takes as input cross sections (or level sets) of an object as well as the number of handles the object is known to have. The edges of the Reeb graph are generated as the areas where the contours do not change. Edges at saddle points which correspond to contours which are close together are connected to complete the Reeb graph by making sure not to contradict the known number of handles. The running time of the algorithm is $O(n^2)$ where n is the number of edges in the triangulation. An improvement on the running time of this algorithm was the suggested in Hilaga *et al.* (2001a) by introducing Multiresolution Reeb graphs. The key idea proposed in the paper is to sacrifice accuracy in order to speed up computation by computing a quantised version of the Reeb graph at various levels of details. A more efficient algorithm for Reeb graphs of 2-manifolds was given in Cole-McLaughlin *et al.* (2004). The algorithms uses a result from Morse Theory which states that changes of the topology of level sets can only occur at the critical points of the input mesh. Since these can be recognised via a local test for criticality the algorithm first finds all critical points and sorts them in descending order. The algorithms then proceeds to create, destroy, cut and glue together components of level sets as it sweeps through them. The resulting running time is $O(nlog(n))$ where n is the number of edges in the triangulation.

The first on-line Reeb graph computation Pascucci *et al.* (2007) was introduced in 2007. While its running time is quadratic, it can handle arbitrary meshes, it has small memory footprint and good performance in practice (especially on 2D meshes). The algorithm works by dynamically creating and updating the Reeb graph as it reads the input of vertices, edges and faces in a stream. Note that we do not need to read the higher dimensional simplices because the Reeb graph only depends on the 2-skeleton of the domain Pascucci *et al.* (2007). For each new vertex a new node is created in the Reeb graph and for each new triangle we add a new edge. Adding the face of a triangle then connects disjoint contours and corresponds to merging two paths of the Reeb graph. In the end of the computation all degree two vertices are removed by merging their adjacent arcs. In 2009 Tierny et. al Tierny *et al.* (2009) presented an algorithm that takes as input 3-manifolds with boundary, embeded in three dimensional space. While it is not general, the algorithm is efficient with running time $O(nlogn + hn)$ where h is the number of loops in the Reeb graph. The algorithm introduces the key concept of loop surgery and decomposes the domain into simply connected regions where the Reeb graph has no loops. Reeb graphs of such regions are a special case that is called the contour tree because they are connected and acyclic. Unlike with the Reeb graph there are general case efficient algorithm for contour tree computation with running time of $O(nlogn)$ Carr *et al.* (2000) that work by computing and combining the connectvity of the sublevel and the superlevel sets (see Section 3.1) of the domain. The contour trees of the simply connected regions are joined together at the point of surgery to create the loops of the Reeb graph.

Consequently Doraiswamy and Natarajan produce a number of papers Doraiswamy & Natarajan (2009, 2012, 2013). Their first algorithm borrows the idea of tracking level set components during an isovalued sweep of the input mesh Doraiswamy & Natarajan (2009). They adapt it to the three dimensional case by computing the Reeb graph incrementally and use a more complex tracking procedure. The running time for 3-manifolds is $O(nlogn+nlog(g)(log(log(g)))^3)$ and $O(nlogn(log(logn))^3)$ for d-manifolds where g is the maximum number of loops in any level set. Their following paper Doraiswamy & Natarajan (2012) is based on an alternative definition of the Reeb graph where nodes and arcs are mapped to components of critical level sets and equivalence classes of regular level sets respectively. As with the previous algorithm the critical points in the domain are located via a local criticality test and sorted by value. The algorithm uses the alternative definition to identify pairs of critical points that have a cylinder between them and inserts the corresponding nodes and arc in the Reeb graph. The overall running time of this algorithm is $O(n + m + tlogt)$ where m is the number of critical points and t is the size of the critical level sets. Their final algorithm uses an idea similar to loop surgery Tierny *et al.* (2009) to divide the input into subvolumes whose Reeb graphs are loop free and then computes and combines contour trees to obtain the final Reeb graph. The improvement they have made is that their approach applies to simplicial complexes of any number of dimensions. The overall running time is $O(mlogm + sn)$ where m is the number of vertices, n is the number of triangles and s is the number of saddles.

The first two near optimal algorithms are Harvey *et al.* (2010); Parsa (2013). The first algorithm Harvey *et al.* (2010) is randomised with expected running time of $O(mlogn)$ where m is the size of the 2-skeleton and n is the number of edges. The algorithm works by iteratively collapsing triangles in the input mesh until the Reeb graph is obtained. Collapsing all triangles around a vertex is a systematic way to collapse the entire contour that passes through that vertex. This operation is performed on all vertices which are first randomly permuted. The second algorithm Parsa (2013) is deterministic and has running time of $O(mlogm)$ where m is the size of the 2-skeleton. It works in two phases; first it detects and sorts all critical vertices based on their value; then it reduces the problem to maintaining connected components of a graph through insertion and deletion of arcs. The authors use domain specific knowledge to solve that problem with optimal timebounds using the offline graph connectivity problem Eppstein (1994). Furthermore they go as far as to show the inverse reduction from the offline graph problem to the Reeb graph construction, thus showing that the two computations are equivalent. What both algorithms Harvey *et al.* (2010); Parsa (2013) have in common is that they leave the continuous cases behind and work with graphs and abstract simplicial complexes instead of their geometric realisations. This yields combinatorial algorithms that borrow more from graph theory than previous algorithms.

3.4 Contour Tree Algorithms

Often in scientific visualisation the domain is a connected volume. In those cases the Reeb space is also connected and it does not have cycles. This restricted case of the Reeb space is called the contour tree. The contour tree is useful in practise because there are more efficient algorithms to compute it and it applies to many real life data sets.

The standard contour tree algorithm Carr *et al.* (2003) is based on the idea of an isovalued sweep - i.e. processing the vertices of the mesh in sorted order from high to low. As each vertex u is processed, any edge (u, v) to a higher-valued vertex v is also processed. At each step there is a subgraph representing the super-level set, whose connectivity can be tracked with an incremental version of the union-find data structure Tarjan (1975).

In the first stage of the algorithm we construct the join tree, then repeat with a low-to-high sweep to compute the split tree. In the second stage, we construct the contour tree iteratively by transferring leaves and their adjacent edge from the merge trees, using induction on a simple invariant to guarantee correctness. As a result, this algorithm is sometimes referred to as the sweep and merge algorithm.

There have been several approaches to scaling the sweep and merge algorithm. Some of them have focused on distributed computation Landge *et al.* (2014); Morozov & Weber (2013, 2014); Pascucci & Cole-McLaughlin (2004). We will not consider them because they focus more on minimising communication between nodes than efficient utilisation of individual nodes. Other methods focus on vector Carr *et al.* (2016), thread

Gueunet *et al.* (2016); Gueunet *et al.* (2017) or hybrid Acharya & Natarajan (2015); Maadasamy *et al.* (2012); Rosen *et al.* (2018) shared memory parallelism.

The parallel peak pruning algorithm Carr *et al.* (2016) is the only shared memory algorithm which parallelises the merge phase. Other algorithms introduce a novel way of computing the join and split tree, but combine them in serial. Note that the merge phase has linear complexity and it is faster to compute than the join and split trees. Nonetheless, parallelising the merge phase is important for resource utilisation and parallel speed up according to Amdahl's law Gueunet *et al.* (2017).

In the merge phase of the serial contour tree algorithm Carr *et al.* (2003) we transfer the leaves and their adjacent edge from the merge trees to the contour tree. Since this is a local operation all leaves can be batched and transferred in a single parallel step. The algorithm alternates between transferring leaves from the join and split tree until the contour tree is fully constructed. In the ideal case, each batch transfers at least half of the vertices, guaranteeing logarithmic performance. In a tree with no vertices with degree two half of the vertices are leaves. Thus we can achieve logarithmic collapse if we remove degree two vertices in a post process for each batch.

Removing a degree two vertex is straightforward when its neighbouring vertices have values spanning the value of the vertex. This is the case when the vertex is connected by a chain of such vertices to a leaf. In effect, these vertices are regular at this stage (although they may not have been in earlier stages), and can be removed. For vertices of degree two whose value is smaller or bigger than the value of both neighbours, this is not so easy to perform. We call these vertices forks.

A W-structure consists of repeated forks zigzagging between upwards and downwards, as illustrated Hristov & Carr (2019). In order to collapse the W-structure completely we can only prune from an endpoint of the W-structure to a fork. The internal vertices cannot be process until we have pruned all forks. Therefore computation is effectively serialized along the largest W-structure in the contour tree. This prevents logarithmic collapse and complicates the parallel complexity analysis of the algorithm.

3.5 Topological Simplification

Simplification is motivated by the practical need to deal with noise in real life data. In other fields like scientific computation, image processing and geometry processing simplification is better known as data denoising or smoothing. Essentially simplification is used to remove things such as statistical noise and measurement error in order to highlight important features. Topological simplification specifically refers to a simplification technique that removes small scale topological features without interfering with large scale ones. Methods for topological simplification usually remove critical points which in turn simplifies topological features. These methods either simplify the function directly, which in turn simplifies the topological structures derived from it; or they simplify the topological structures and they in turn correspond to a simplified function on the domain.

First, we will look at techniques that simplify the input function. Morse theory

provides a method for cancelling critical points in pairs via the Morse Cancellation Theorems Matsumoto (2002b). The Morse Cancellation Theorems state that for certain pairs of critical points we can perturb the input function slightly in the region around the critical points to obtain a simpler Morse function that contains all critical points except for the chosen pair. An approach that simplifies critical points in pairs directly for scalar fieds is Lukasczyk *et al.* (2021).

Since in the theory behind critical point cacellation uses the flow of gradient-like vector fields in practise this can be done using the Morse-Smale complex. A number of methods take this approach Bremer *et al.* (2004); Gyulassy & Natarajan (2005); Günther *et al.* (2014); Jacobson *et al.* (2012); Ni *et al.* (2004); Weinkauf *et al.* (2010). They compute the Morse-Smale complex, find critical points whose index differs by one and are connected by a common arc and then apply a simplification based on topological persistence or some other geometric measure like volume or surface area. One issue with this approach is that it does not directly give rise to a simplified function, but to a simplified Morse-Smale complex and a simplified gradient vector field. This is why these methods usually involve some sort of numerical computation like nonlinear optimization Günther *et al.* (2014); Jacobson *et al.* (2012); Ni *et al.* (2004); Weinkauf *et al.* (2010) or solving partial differential equations to apply partial Laplacian smoothing Bremer *et al.* (2004) to simplify the actual input function. These methods are computationally intensive and lead to issues with numerical instability. An alternative combinatorial approach that does not rely on the Morse-Smale complex is referred to as ϵ-closeness Edelsbrunner *et al.* (2006). It uses Persistent homology to produce a function whose distance from the original is no more than ϵ and has no pairs of critical points with persistence less than ϵ. This work was further improved by Attali *et al.* (2009); Bauer *et al.* (2012) and a more general scheme that does not rely on persistent homology is given by Tierny & Pascucci (2012).

The direct approach on the other hand applies simplification to the topological structure directly. There are several similar techniques for contour tree simplification which are based on pairing maxima and minima with saddle points. One technique is called Leaf pruning Carr *et al.* (2010a) and it involves removing a leaf from the tree and reducing the saddle it connects to to a regular point. This technique is always applicable because a tree has at least one more leaf edge than it has interior points. A similar technique to this is Branch decomposition Pascucci *et al.* (2004b). It involves partitioning the contour tree into monotone paths called branches and then removing those that do not disconnect the tree in order of persistence. We must note that when these methods use persistence, their output is not the same as that of persistent homology as show by Petar Hristov Hristov (2017). In the general case, these techniques cannot directly be applied to the Reeb graph because we are not always guaranteed to have leafs due to the presence of cycles. The only available technique to handle this is extended persistence Agarwal *et al.* (2006) which pairs all points on a closed manifold. It pairs minima and maxima to saddles, and merge saddles to split saddles which correspond to loops in the Reeb graph. Pairs of minima, maxima and saddles can be removed similarly to leaf pruning and loops can be cancelled by gluing the

two paths of the loop and eliminating their endpoints as they become regular points Pascucci *et al.* (2007).

3.6 Multvariate Field Methods

One of the first methods for comparing multiple scalar fields was introduced in Hilaga *et al.* (2001b). The authors use a coarse representation of the Reeb graphs of different surfaces to compare them by determining an index of similarity. This coarse representation is based on computing multiple quantised versions of the Reeb graph jointly called the Multi-Resolution Reeb graph. The method is implemented as a comparison function in a 3D geometric shape database with satisfactory results. This approach was extended by Zhang *et al.* (2004) to comparing scalar functions of simply connected three dimensional domains. The authors compute a quantised approximation of the contour tree at multiple resolutions similar to Hilaga *et al.* (2001b) and apply a graph matching algorithm that directly compares the similarity of the attributes of the contour tree nodes. They use their method to compare proteins using their electron density and electrostatic potential. An alternative to computing a quantised version of the Reeb graph and Contour tree is given by Schneider *et al.* (2008a). The authors propose to compute and simplify the Contour trees of two scalar fields use them to construct the two largest contour segmentations Manders *et al.* (1996) where the largest contour segmentation is defined as the largest surface containing only one local maximum. The next step is to compute the spacial overlap of the largest contours of both scalar fields and present them as nodes in a weighted bipartite graph where the weights of the edges represent the similarity of the overlapping contours. To enable interactive exploration, the authors provide a linked view of the contour trees and the domain where the user can select contour pairs form the Contour tree and study their relationship in the domain.

While these papers have shown that Reeb graphs and contour trees can be used for comparing scalar fields, they merely identify similarities between individual fields rather than giving the overall structure of their topological correlation. A more advanced idea is that of layered Reeb graphs Strodthoff & Jüttler (2015). Layered Reeb graphs were first defined in Strodthoff & Jüttler (2013) for two Morse functions f and g on a common d-manifold. The idea is to compute the Reeb graph of the first function and then the Reeb graphs of the second function, restricted to the level sets of the first function. The authors do not compute the Reeb graphs of all level sets, only at isovalues where their topology changes. As we will see in Secition 3.8 the change in topology happens in level sets which cross a generalisation of critical points called Jacobi edges. To visualise this construction, the multiple Reeb graphs of the second function are displayed next to the parts of the edges of the Reeb graph of the first function where all level sets have the same Reeb graph.

Another set of approaches compare the gradients at every point in the domain and compute an index of correlation. In the first approach Nagaraj *et al.* (2011) the authors compute the biggest length of any unit vector multiplied by the Jacobian.

This induces a new scalar field which provides information about the interactions of the different fields with one another. It has been applied to data from climate science and combustion simulations. There are two main issues with the approach; the first one is that it is sensitive to scaling of the fields and the second is that it is difficult to identify which isovalue of the measure is useful for producing isosurfaces. The second method is used for visualising correlations in 3D scalar data fields Sauber *et al.* (2006). The authors compute and visualise the amount of minimum local correlation at every point in the data set. One of the main drawback of this method is that if two fields are not correlated we lose information about all other fields. Another drawbacks is that correlation is a linear measure, so it may not detect complex non-linear relationships. A similar approach has been studied in Gosink *et al.* (2007), as the authors compute the dot product of the gradients of the fields. However this method can only be used for comparing two scalar fields at a time. An issue with all of these methods is that they are too reductive. That is, they describe the local relationship of the interaction of multiple fields using a correlation indicator. We argue that more complex topological structures are needed to give a more complete and detailed description. In Section 3.8 we will describe how we can use gradients of the functions to obtain richer information about the topological correlation of scalar fields using the Jacobi sets.

3.7 Fiber Surfaces

Fiber surfaces Carr *et al.* (2015) extend the notion of isosufaces by considering two scalar functions f_1 and f_2 combined in a single bivariate function $f : \mathbb{R}^3 \to \mathbb{R}^2$. The inverse image of a point (x, y) is well defined as a fiber $f^{-1}(x, y) = f_1^{-1}(x) \cap f_2^{-1}(x)$. A fiber is the intersection of two isosurfaces, or a one dimensional curve in the domain. In order to obtain a surface instead of a curve we further extend fibers to fiber surfaces by taking the inverse image of a path instead of a single point. This way the fibers of every point on the path sweep out a set of continuous surfaces. Fiber surfaces share two important properties with isosurfaces. One is that each connected component of a fibre surface is a continuous surface. This guarantees a coherent visualisation. The second one is that the Fiber surface of a path, which disconnects the range, disconnects the domain. This enables enables an algorithm for efficient computation.

Fiber surface computation as described in Carr *et al.* (2015) is an extension of the Marching Cubes algorithm. The input is a regular rectilinear mesh where data values are sampled at the vertices and interpolated otherwise and a closed path in the range that is approximated by a polygon. For the purposes of this computation we will refer to this polygon as a Fiber surface Control Polygon (FSCP). The first step in the computation is to label all the vertices in the cubes of the mesh as black or white depending on whether they are mapped to the inside or the outside of the FSCP. We then use the cases provided by Marching Cubes so long as we find the point of intersection between the Fiber surface and the edges of the cubes. This can be done with a line intersection test of the projection of the edge in the range and the FSCP. Instead of depending on slow geometric intersection tests we can speed up computation

by observing that the Fiber surface for the FSCP is the zero level set of the FSCP's signed distance field. We can then take an isosurface from this derived scalar field.

There have been several improvements and extensions to this approach. Firstly we will discuss issues with the original algorithm we just described Klacansky *et al.* (2017a). They come in part from the underlying Marching Cubes algorithm and from the nontrivial details of how the signed distance field of the FSCP is constructed. In practise the signed distance field is computed on a rasterisation of the range. Even for large resolutions of the rasterisation, the Fiber surface can still fail to capture small details and sharp features. Another issue is that vertex classification fails when the image of the tetrahedron is contained in the FSCP. This incorrectly produces an empty zero level set. Finally in the case where the FSCP is not convex, it may cross the images of the edge multiple times. This causes incorrect classification which leads to inaccurate topology and can even miss out some connected components of the Fiber surface.

To solve these problem the authors of the paper Klacansky *et al.* (2017a) introduce an extension of marching tetrahedra for Fiber surfaces. To compute fibers the authors intersect two isosurfaces with respect to the two components of the field. In the case of hexahedral cells with trilinear interpolation fibers become intersection of hyperbolic sheets. This arbitrary complexity makes adapting this approach impractical so the authors use tetrahedra instead. Since barycentric interpolation is assumed, the isosurfaces are the intersections of parallel planes in the tetrahedron. The intersection two isosurfaces corresponding to a fiber therefore results in a line segment. Consequently any two fibers in a tetrahedrons are co-planar and parallel within that plane. To compute a fiber in practise we extract the isosurface of the first component function using Marching Tetrahedra, then interpolate the value of the second function and use marching triangles to extract the fiber. To compute the Fiber surfaces we split the FSCP into line segments, compute their Fiber surface and clip them at the endpoints of the line. The Fiber surface of each line is computed as a derived scalar field using signed distance and the normal form of a line $n.p = c$, where n is a normal vector to the line and c is the distance to the line scaled by the length of n. To clip the Fiber surface at the endpoints of the line segment we determine whether the endpoints of the triangle that is the intersection of the tetrahedron with the zero level set are between the clip points or not. The algorithm inherits its running time and available parallelism from the underlying marching tetrahedra algorithm. One of the advantages of using this algorithm is that the fibers and Fiber surfaces of the line segments from the FSCP can be textured using different colours. This enables better visual analytics by identifying points on the FSCP with their corresponding fiber in the domain.

Another improvement is a generalisation to any number of dimensions for the range and the domain with the paper Jankowai & Hotz (2018a). The paper studies functions $f : \mathbb{R}^n \to \mathbb{R}^m$. It defines traits as geometries or sets of points in the range and feature level sets as their preimages. These two concepts generalise FSCP and Fiber surfaces respectively. Feature level sets are computed as the isosurface of the zero level set of the trait's distance field. The paper not only introduces this generalisation, but it also

provides an implementation in a workflow as well as a suggestion for a visualisation system. The visualisation system consists of a trait design interface and a linked view. Picking traits requires an appropriate visualisation of the range where potential traits are identifiable using domain specific knowledge. The linked view allows the user to pick traits from the range in real time and get the feature level set as feedback. This interactivity enables dynamic exploration of the data set. The authors demonstrate this by applying their system to numerical simulations of solids, vortices and hurricanes.

Lastly we present the application of direct volume ray casting to Fiber surfaces Wu *et al.* (2017). Volume ray casting offers a number of benefits when applied to Fiber surfaces. One is that volume data can be rendered with pixel-exact accuracy which allows for correct rendering of details and sharp features. The other one is that since it does not extract any geometry it is fast and efficient and enables real time interactive exploration of the data set. Rays are intersected with the Fiber surface by projecting them into the range and determining the point of intersection with the FSCP. To speed this up the authors compute the signed distance field of the FSCP and find intersections by interpolating between points in the ray which have opposite signs. One limitation of this approach is that if there are multiple FSCP their distance fields cannot be merged, they have to be computed and tested for intersection separately. Another issue is that the method assumes that there is only one intersection point between the FSCP and the image of the ray and that causes some rays to miss the Fiber surface if their sampling step is too big.

In practise polygons in the range are not chosen blindly. One approach is to use a continuous scatterplot to project the data to the plane Bachthaler & Weiskopf (2008a). We can then pick out areas of interest in the scatterplot and enclose them in a FSCP to extract the corresponding area in the domain. The continuous scatterplot allows us to visualise how the two functions are related in the codomain and the Fiber surface allows us to see how they are related in the domain. This allows for better understanding of the structure and relationship of the two scalar fields. One interesting application of Fiber surfaces is in the study of Fiber Topology by mathematicians. In this example a system was create to help identify and visualise critical fibers Sakurai *et al.* (2016). Another application is to covalent and non-covalent interaction within molecules Carr *et al.* (2015) as well as combustion simulation Wu *et al.* (2017). An implementation of fiber surfaces is included in the TTK plugin for Paraview Tierny *et al.* (2018).

3.8 Jacobi Sets

Jacobi sets Edelsbrunner & Harer (2002a) are the first topological insight into the correlation of the components of multivariate functions. They extend the notions of critical points for $k > 1$ Morse function simultaneously defined on a common d-manifold. In the case of two functions, the Jacobi set is the set of critical points of one function restricted to the level sets of the other. The definition is symmetric so it does not matter which function we take the level sets of. An equivalent definition is one where the Jacobi set is the set where the gradients of the two functions are linearly dependent.

This results in a smoothly embeded 1-manifold in domain, or a subset of the edges for a combinatorial manifold. It partitions the domain into regions based on the relative orientation of the gradients of the two functions. An important note is that the Jacobi set is well defined only when the number of functions is less than or equal to the dimension of the domain.

The algorithm for Jacobi set computation Edelsbrunner & Harer (2002a) relies on a local test for determining whether an edge is critical or not. The local test for criticality is based on computing the Betti numbers of the lower link of the vertices incident to the edge. Once the critical edges are found they are assembled together to form the Jacobi sets. In the general case of $k > 2$ functions we test every $k - 1$ simplex for criticality by computing the Betti numbers of its lower link. This algorithm is efficient only when the dimension of the domain is less than 5. To see why we first note that since the domain is assumed to be a triangulation of a d-manifold the link of a vertex is in turn a triangulation of the $d-1$ sphere Edelsbrunner *et al.* (2008b). This is the combinatorial of equivalent of the manifold property - every point on the manifold has a neighbourhood that is isomorphic to $\mathbb{R}^d$. There are efficient algorithms for computing the Betti numbers of simplicial complexes on the 3-sphere Delfinado & Edelsbrunner (1995), but none in the general case. In higher dimensions, using homology with coefficient modulo two, the computation is equivalent to Gaussian elimination on the incident matrices of the lower stars. Another way to compute the Jacobi set of meshes and points clouds is to approximate the gradients Luo *et al.* (2009).

The mathematical theory behind Jacobi sets is not fully understood in the general case. Jacobi sets are related to the study of singularities in a generalisation of Morse Theory that is called Singularity Theory Bhatia *et al.* (2015). Singularity Theory extends Morse Theory and studies functions to higher dimensional manifolds. One of the major results in Morse Theory is the Morse lemma which allows us to completely understand the differential topological behaviour of a function at critical points. This leads to a characterisation of different types of critical points - local minima, local maxima and different kinds of saddles. A consequence of this is that the topology of sublevel and level sets only changes at critical points. In singularity theory we consider pairs (m, n), where m is the dimension of the manifold domain and n is the dimension of the codomain, usually $\mathbb{R}^n$. There are no strong results that are similar to the Morse Lemma in the general case, but there are certain results for low dimensional pairs. We can generalise the notion of Morse functions and critical points in low dimensional cases where $n = 2, 3$ to stable functions and critical fibers and we can characterise the singular fibers in the $(3, 2)$ and $(4, 3)$ cases. Higher dimensional cases are an active area of mathematical research.

Singularity theory has provided valuable valuable information about the classification and computation of singularities and singular fibers for use in visualisation and computational topology. On the other hand multivariate visualisation methods have been applied by specialists in singularity theory to better understand $\mathbb{R}^3 \to \mathbb{R}^2$ functions Sakurai *et al.* (2016). The authors develop an application that interactively visualises fiber singularities to enable analytical investigation through manual visualisation. The

user of the application iteratively selects fibers to visualise until they have explored the fiber topology of the function. Then they perturb the function and repeat the process to gain understand how the topology has changed. Jacobi sets have found other practical applications in feature tracking of combustion analysis Bremer *et al.* (2007); estimation of interrelationships between geophysical multifields Artamonova *et al.* (2017); combinatorial ridge detection Norgard & Bremer (2013); multivariate topological simplification Bhatia *et al.* (2015); N & Natarajan (2011) and generating a map between triangulated meshes of arbitrary genus Bennett *et al.* (2007). Open research questions in this area are how we can simplify and compute Jacobi sets efficiently in the general case, how we can visualise them in higher dimensions and how they can be used to interpret data.

3.9 Reeb Space

The Reeb space Edelsbrunner *et al.* (2008b) is a generalisation of the Reeb graph in the case where two or more scalar functions are defined on a common domain. Analogously to the Reeb graph, the Reeb space is constructed by contracting the connected components of all fibers to a single point. Unlike the Reeb graph, the Reeb space does not even have to be manifold and its Betti numbers can be arbitrarily large compared to the domain Basu *et al.* (2018). This makes the Reeb space difficult to interpret and understand. However, for certain functions the Betti Numbers are bounded by the algebraic complexity of the function and we can decompose the Reeb space into multiple disjoint manifolds that are glued together in possibly complicated ways (this is called a stratification). A similar construction to the Reeb space, called the Stein Factorisation, is studied in Singularity Theory Burlet (1974). For $f : M \to N$ a map between two manifolds we define W_f as the quotient space of connected components of fibers. We also define the quotient map $q_f : M \to W_f$ and the natural quotient map and we define $n_f : W_f \to N$ such that $f = n_f \circ q_f$. The Stein Factorisation is used to the study the function and its singularities as well as the topology of the domain. A number of important results have been obtained for the Stein Factorisation Emeristo. Mata-Lorenzo (1986). For example the Stein Factorisation of topologically stable maps are triangulatable, and thus their quotient spaces are polyhedrons. This fundamental result enables us to study the Reeb space by analysing the polyhedral structures in it.

The triangulation and stratification of the Reeb space of a combinatorial manifold equipped with k generic piecewise linear functions can be computed in polynomial time Edelsbrunner *et al.* (2008b). The first step of the computation is to use the $k + 1$ skeleton of the combinatorial manifold to decompose the domain into polytopes which are aligned with the fiber of the functions. This is analogous to keeping track of sublevel sets in the Reeb graph computation. Those aligned subsets are then merged together in the cases where topological change does not occur or glued together along their shared faces when topological change does occur. Once we triangulate the merged and glued together polytopes we readily obtain the complex that is the Reeb space. To stratify the Reeb space we group simplicies together to form manifolds. To do this

we first take the simplices of dimension k as separate pieces since they are already manifold. We then add simplicies of lower dimension to merge and enlarge those pieces as long as they remain manifold. The routine that determines whether two pieces can be merged relies on a subroutine that determines whether two triangulated spaces are homeomorphic. This problem is known to be undecidable for dimensions bigger than four Markov (1958). The authors do not provide an implementation or an asymptotic analysis of the running of the algorithm.

The first practical Reeb space algorithm was given in the special case of a bivariate function defined on a 3-manifold Tierny & Carr (2017a). The algorithm relies on the property that topological changes in the fibers of the function can only occur near the Jacobi sets of the domain. The authors project the Jacobi Edges to the range and then construct the so called Jacobi Fiber surfaces as the preimage of the image of the Jacobi set, using know methods from Fiber surface computation. The Jacobi Fiber surfaces segment the domain into regions and allow us to compute the Reeb space by taking the connected components of the segmented domain and gluing them together where they have an adjacent Jacobi Fiber surface. The components of the Reeb space are two dimensional sheets glued along one dimensional edges. The Reeb space is then presented visually with a novel technique called Scatterplot Peeling. With this technique the components of the Reeb space are layered on top of the continuous scatterplot of the data and the user is allowed to select sheets of the Reeb space for inspection in decreasing order of projection area. The algorithm is trivially parallelisable and has quadratic running time due to the Jacobi Fiber surface computation. In addition to this, there are two approximate algorithms that construct structures which converge to the Reeb space. They are called the Join Contour Net and Mapper and we will discuss them in more detail in Section 3.10 and Section 3.11.

As we have already seen, the Jacobi sets is where the topological change of fiber happens. However the relationship between the Jacobi set and the Reeb space is harder to understand. A first step towards understanding that relationship was made by introducing Jacobi Structures Chattopadhyay *et al.* (2014). Jacobi Structures are the projection of the Jacobi set into the Reeb space. They are equivalent to the critical nodes in a Reeb graph. We can use them to parametrise the singular fibers of the domain to detect where topological change happens in the Reeb space. The major open areas of research regarding Reeb spaces are in developing efficient algorithms for computation and simplification and developing understanding of how the Reeb space can be used in identify and visualise correlations and the topological structure of multivariate functions.

3.10 Mapper

Mapper Singh *et al.* (2007) is a discrete representation of the high dimensional topological structure of a space. In order to construct Mapper we need to define a function f on the domain and a number of open sets $\{U_1, U_2, ..., U_n\}$ whose union is equal to the codomain of f. In the terminology of this method the function is called a filter and

the open sets are called a cover. Mapper is the simplicial complex that describes the connectivity of the preimages of these open sets $\{f^{-1}(U_1), f^{-1}(U_2), ..., f^{-1}(U_n)\}$. The vertices of Mapper are the set $\{1, 2, 3, ..., n\}$ and it contains the k dimensional simplex $\{i_1, i_2, ..., i_k\}$ whenever $f^{-1}(U_{i_1}) \cap f^{-1}(U_{i_2}) \cap ... \cap f^{-1}(U_{i_k}) \neq \emptyset$. This general idea is put into practise by analysing point cloud domains via Eucledian codomains. The goal is to understand the topological structure of the underlying continuous space that we assume the point cloud is sampled from via the function and the decomposition of the codomain. The topological notion of connected components does not make sense for point clouds, so the authors substitute it with the statistical notion of clustering. While the authors do no explicitly derive the computational complexity of Mapper we can observe that it depends on the number of samples in the point cloud, the number of sets in the open cover of the codomain and on the complexity of the chosen clustering method. The relation of Mapper to the Reeb space can be explained when we consider their categorical representation. The image of the categorical Mapper converges to the image of the categorical Reeb space for increasingly refined covers Munch & Wang (2015). What this means in practise is that Mapper is equivalent to the Reeb space for a cover of sufficiently high resolution.

One of the main practical issues with using Mapper is that its output is highly dependent on the choice of filter and cover. While this flexibility allows us to gain insight into specific aspects of the data, recognising which functions and covers work best is highly non trivial. In the worst case one needs to already know something about the topological structure of the domain before making an educated guess. Furthermore, there are no strong guarantees on the stability of Mapper under slight pertubartions of the filter and the cover Cohen-Steiner *et al.* (2007). A different choice of either can make the result of Mapper very different and it is not easy to see how we can reconcile the different results. There have been several notable improvements on the Mapper algorithm. By imposing certain conditions on Mapper we can obtain structural and stability theorems. The structure and stability of one dimensional Mapper has been studied in Carrière & Oudot (2017). The one dimensional case is less useful since in this special case Mapper approximates the Reeb graph for which we have efficient and exact algorithms. Mapper has been used as a clustering method for gene expression analysis Jeitziner *et al.* (2018). The authors call this extension Two-Tier Mapper and use it to obtain stability theorems for their particular choice of filter derived from computing gene deviation. Another improvement is Multiscale Mapper Dey *et al.* (2015). The idea there is to construct multiple covers of the codomain of increasing resolution and to show that the Mapper output with respect to each cover gets refined as a result of the increasing resolution. The authors demonstrate that we can construct a filtration using the outputs of Mapper and use Persistent homology to extract topological information about how Mapper changes with cover refinement. Another important result from that work is that Mapper may not admit a simple notion of stability such as the one that researchers have obtained for other tools like the Reeb graph and Persistent homology. Other notable improvements of Mapper give insight into parameter selection Carrière *et al.* (2018); present a similar construction, based on Vietoris-Ripps complexes, that is

parameter free Liu *et al.* (2012); an algorithm for distributed computation Hajij *et al.* (2017) and a study on the one dimensional homology of Mapper Dey *et al.* (2017).

Mapper has been used to identify a subgroup of breast cancers with a unique mutation profile and excellent survival rate Nicolau *et al.* (2011); to detect patterns that predict long-term recovery for various drugs in preclinical spinal cord injury and traumatic brain injury Nielson *et al.* (2015); to characterize transient cellular states in RNA-seq analysis Rizvi *et al.* (2017); to visualise the overall organization of whole-brain activity maps Saggar *et al.* (2018) and to analyse gene fusion across different tumor types Frattini *et al.* (2018). Other applications include simulation data for biomolecular folding pathways Yao *et al.* (2009), head and neck cancer subtypes De Cecco *et al.* (2015), mesoscale network dynamics in neuroscience Khambhati *et al.* (2018), detecting divergent subpopulations in phenomics data Kamruzzaman *et al.* (2018), heart rate data Carlsson *et al.* (2009a), network analysis Coudriau *et al.* (2016); Khambhati *et al.* (2017) and biomechanical gait analysis Phinyomark *et al.* (2018).

3.11 Join Contour Net

The Joint Contour Net is a quantised generalisation of the Reeb graph Carr & Duke (2014). It expresses the connectivity of regions with common properties with respect to multiple parameter functions. In JCN terminology a join level sets is the preimage not of a single point, but of a quantised area around the point, which is equivalent to rounding. A slab is then a single connected component similar to a contour, and a fragment is a part of a slab inside a single cell from the domain. To construct the JCN we first construct the Joint Contour Graph. In this graph vertices correspond to fragments and an edge between two fragments indicates that they are adjacent via a cell boundary. Upon contracting all fragments of the same joint level set, we effectively contract all slabs and obtain the Joint Contour Net. The overall running time of the algorithm is $O(rN_e+N_e\alpha(N_e))$ where r is the dimension of the range and N_e depends on the product of the levels of quantisation for each function and the number of simplices and α is the inverse Ackermann function. In practise this upper bound is shown to rarely be achieved because most cells only have a few quantisation levels.

The Join Contour Net is a special case of the more general Mapper where the domain is a simplicial mesh as opposed to a point cloud. The quantisation can be seen as a partition of the range and instead of clustering the preimage of the quantised cells we are able to directly obtain the connected components geometrically because the domain is a simplicial mesh instead of a point cloud. The convergence of Mapper to the Reeb space also applies to the JCN. The Joint Contour Net is computable even the number of functions exeeds the dimension of the domain. In that case the JCN retessellates the function with respect to range properties instead of domain properties.

The Joint Contour Net has been applied in practise to visualising nuclear scission, nuclear fission, hurricane data and lattice quantum chromodynamics Thomas *et al.* (2017). In Duke *et al.* (2012) the JCN was used to detect the spontaneous fission of fermium nuclei and identify that scision is not a single combinatorial event, but a

process that occupies a whole region within collective space. The paper Thomas *et al.* (2017) on the other hand presents a potential use of the JCN in Lattice Quantum Chromodynamics which are used by physicists to model strong nuclear force. The JCN is used as a compact descriptor of data sets which are so large as to prohibit manual visual inspection. The paper determines that some patterns observed that using persistence derived from the JCN corresponds to physical observations.

3.12 Pareto sets

Pareto optimality was initially developed by Vilfredo Pareto in the 19th century Rossi (1840) in the field of multiobjective optimisation Miettinen (2012). The idea was further developed and found applications in Optimisation Theory, Statistics and Genetic Programming. It was later applied Huettenberger *et al.* (2013) in the multivariate piecewise linear setting as a method to identify regions in the domain where all the functions "agree" on the ascending and descending paths from a point. More formally, we say that points in the input mesh are comparable when all scalar functions either increase or decrease when we go from one point to the other. When they all increase we say x dominates y and when they all decrease we say that x is dominated by y. Regions of the mesh where all points are incomparable are called Pareto optimal, points that have a neighbourhood such that all points in that neighborhood are either incomparable or dominate them are called Pareto minima and points with a neighborhood inside which all other points are either incomparable or are dominated by them are called Pareto minima. The Pareto optima, minima and maxima are collectively called Pareto extrema. To compute the Pareto set the authors of Huettenberger *et al.* (2013) use marching triangles and marching tetrahedra for 2 and 3 dimensions respectively to compute ascending and descending sets inside the simplicies. These are the sets of points where all points have a bigger or smaller function value for all functions. The algorithm has worst case complexity of $O(d^n)$ where d is the dimension of the domain and n is the size of the input, however the authors claim that this can be reduced to a solving a linear system to obtain a running time of $O(n^{3.5} \cdot N \cdot d!)$ where N number of simplices of higher order Huettenberger *et al.* (2017a).

An extension of Pareto sets is the Reachability graph Huettenberger *et al.* (2014). The authors extend the idea of ascending and descending paths between simplices to global connections between the Pareto extrema. The resulting structure is a directed graph where nodes are connected components of the pareto set and directed edges indicate the there is a strictly ascending path from one node to another. The Reachability graph can be used to simplify pareto sets by deriving a local comparison measure and reducing the problem to contour tree simplification. A recent paper Huettenberger *et al.* (2017b) demonstrates a relation between the Pareto sets and the Joint Contour Net. It shows that in a directed version of the Joint Contour Net, the Pareto sets appear as critical slabs with both incoming and outgoing edges in the directed JCN. This result only holds in the limit where the JCN converges to the Reeb space and in any practical case the directed JCN can be used to approximate Pareto sets and speed

up their computation. A subsequent paper explores subset and equivalence relations between Jacobi sets and Pareto sets Huettenberger & Garth (2015). Pareto sets are a relatively new concept that has found few practical applications so far including atmospheric vortex visualisation Huettenberger *et al.* (2013) and quality control of series production in car manufacturing Huettenberger *et al.* (2015).

3.13 Multivariate Topological Simplification

In this section we will discuss current methods for simplification of Jacobi sets and Reeb spaces. Jacobi sets are simplified by reducing the number of their components with minimal change to the relationships between the functions. One of the first approaches to Jacobi set simplification Snyder (2004) outlines some the difficulties involved. For one, it is not obvious how one can construct a measure of persistence for Jacobi edges or cycles because once they are discovered by the algorithm they persist and never die. The paper proposes two global metrics for measuring the persistence of components of the Jacobi set. The first metric computes the maximum range interval between any two adjacent vertices within a component. The second metric computes the sum of the distances between the endpoints of all edges in a component. Both metrics are applied to each function separately and the final metric for the components is the maximum of the two. A major issue with this approach is that it can only remove whole components and cannot simplify components themselves. Another problem is that the metrics are computed for each function separately and thus only give indication of the persistence of the components relative to one of the functions. A subsequent approach Bremer *et al.* (2007) that deals with time varying Reeb graphs computes the Morse-Smale complex of the time function in order to pair critical points. It then cancels pairs bellow a persistence threshold and removes small loops in the Jacobi set which lie in successive time steps. While this approach can attribute persistence to Jacobi sets because of the time component, it is limited to time varying data and cannot remove components which span more than one time frame. A more general method is given by N & Natarajan (2011) with a gradient-based measure that computes the cross product of the two gradient vectors. This induces a derived scalar field whose zero level set is the Jacobi set. Simplification is then applied by using the Reeb graph of the derived scalar field to locate and remove components of the Jacobi set in order of the volume. The issue with this approach is that it is not generalisable as it only works for two functions, it can only remove whole components and it cannot remove noise in large components.

A more sophisticated study of Jacobi set simplification is given by Bhatia *et al.* (2015). The authors outline the shortcomings of previous methods for Jacobi simplification and note that applying scalar simplification to two fields independently would not necessarily preserve their joint topological structure. They propose a hybrid method that uses both approaches with a persistence based metric similar to Edelsbrunner *et al.* (2004b). In order to generalise the concept of critical point simplification they modify the first function with respect to the level sets of the other by canceling critical points

in level sets. To avoid creating discontinuities with these cancellations the authors cancel whole connected regions that contain multiple level sets. These regions are then assembled together into sequences of cancellations that ensure that the resulting functions are Morse. The authors demonstrate that for a simply connected domain they can simplify the Jacobi set to a single loop after performing all possible cancellations and they call this the Minimal Jacobi set.

While it is plausible to think that simplifying the Jacobi set would result in a simplified Reeb space, the link between the two has not been formally addressed. A limited method for Reeb space simplification has been introduced in Chattopadhyay *et al.* (2016). This method is an extension of the contour tree leaf pruning procedure Carr *et al.* (2010a) to higher dimensions. It identifies detachable components in the Reeb spaces of simply connected domains called lip components. These detachable components resemble hanging sheets that can be removed without tearing the Reeb space. We are guaranteed that we can simplify these lip components because the Reeb space of a simply connected space is itself simply connected. This however does not guarantee the existence lip components and does not prevent the Reeb space from having 2-cycles or voids. Currently there is no solution to simplifying 2-cycles, like there is for example for simplifying for 1-cycles or loops in Reeb graphs Pascucci *et al.* (2007). Clearly, the simplification of high dimensional Reeb spaces would require a general way of collapsing n-cycles and reattaching their adjacent components back together.

3.14 Time-varying Visualisation and Feature Tracking

Time-varying fields are a key issue in visualisation. The additions of a time component increases the dimensionality of the domain and complicates direct visualisation. For visualisation of time-varying vector fields and flow we refer the reader to Post *et al.* (2004) and for time-varying scalar fields to Lu & Shen (2008); Wang *et al.* (2008); Yu *et al.* (2013). For scalar fields we define features in terms of geometric concepts like isosurfaces or topological concepts like contour trees and Reeb graphs. In extending these approaches to time-varying data we are interested in tracking the development of these features over time. Tracking involves a procedure for deciding whether two features at different time frames are related or not. This relationship structure can be captured by a directed acyclic graph called the tracing graph Samtaney *et al.* (1994).

We will first examine methods which track level sets over time Mascarenhas & Snoeyink (2009) by computing isosurfaces for a given range of values and then varying the time parameter to see how they evolve Samtaney *et al.* (1994). Feature tracking in this case is usually based on volume or special overlap of the interior and exterior of pairs of contours from adjacent time frames Sohn & Bajaj (2006). Some of these methods compute the contour trees of adjacent time steps Szymczak (2005); Widanagamaachchi *et al.* (2012) and find contours corresponding to contour tree edges and how they intersect certain subdomain of interest. Other methods rely on computing merge trees and annotating them with geometric measures that are used to determine correspondence Bremer *et al.* (2011) While the output of these algorithms can be captured

by tracing graphs, each tracing graph is limited to single threshold for the level sets. This means that picking a new threshold requires recomputing the whole tracking procedure. The authors of Lukasczyk *et al.* (2017) propose a nested tracking graph as a hierarchy of tracking graphs, based on the observation that level sets are nested inside each other, in order to avoid recomputing for different thresholds.

A different class of methods track critical points over time. Jacobi sets are used in Bremer *et al.* (2007) to study porous materials and simplification of time-varying data. Another method Reininghaus *et al.* (2012) uses Discrete Morse Theory to track critical points in two dimensional time-varying scalar fields by transforming adjacent time slices into a 3D combinatorial vector field and computing critical lines between the critical points of each time slice. This reduces the problem of tracking critical points to a path search. Persistent homology is applied to tracking critical points Cohen-Steiner *et al.* (2006) by showing that that smooth changes in the function imply smooth changes to the persistence diagrams. The authors implement this as a technique called vineyards and apply it to the study of protein folding trajectories. A more recent method Soler *et al.* (2018) computes Persistence homology pairs of all time frames and then uses the classical optimisation problem of assigning workers to tasks to select correspondence between pairs of critical points.

More advanced methods track the change not only in critical points but in Contour trees and Reeb graphs. This line of research was initiated with a method that uses Jacobi sets to extend Reeb graphs to varying domains Edelsbrunner *et al.* (2004a). Given a scalar function on a time-varying domain we define a new function whose output is the time component. With this definition the preimages of the second function are all points in the domain which have a constant time component. If we compute the Reeb graph of the restriction of the original function to the level sets of the second function we will obtain the Reeb graph of every time frame. Using the two functions defined on the domain we can then compute the Jacobi set which will form a continuous curve that goes through the critical points of the Reeb graphs. This construction is used to track the birth and death of nodes in the Reeb graphs as time varies, which corresponds to the appearance and disappearance of topological features. According to Mascarenhas & Snoeyink (2005) this algorithm is difficult to implement and it requires simplification in order to be practical. The method was later extended with heuristics for fast isosurface extraction and annotations on the Reeb graph of the changes in homology Edelsbrunner *et al.* (2008a). Another improvement Keller & Bertram (2007) tracks topological handles to accomplish smooth topological transitions. A more recent variation on this method Oesterling *et al.* (2017) computes time-varying merge trees instead. This avoids complicated cases and makes the method applicable in arbitrary number of dimensions.

3.15 Scientific Visualisation in Atmosphere Science

Atmosphere sample is a data intensive field that relies on large scale numerical simulations for making weather predictions and evaluating climate models. Visualisation is

a key part of the workflow of the domain scientists, but it is usually limited to simple methods such as diagrams, function plots, scatterplots and other statistical plots. Although 3D visualisation software tools like Vapor Norton & Clyne (2012) and Paraview Ayachit (2015) are sometimes used, their applicability is limited and most visualisation solutions tend to be custom, highly specific and limited in scope. In this section we will focus on feature-based visualisation in meteorology and outline the application of topology based methods in the field. For a comprehensive survey on the subject we refer the reader to Rautenhaus *et al.* (2018) and Tominski *et al.* (2011).

An important topic in Meteorology is cloud formation and evolution. It requires the identification of clouds as features and tracking them through a simulation or a number of observations. Methods in the literature related to this are flow visualisation techniques Post *et al.* (2004), tracking clouds in VR Griffith *et al.* (2005); Heus *et al.* (2009a,b), as well vortex detection Kasten *et al.* (2012); Orf *et al.* (2007) and the generalisation of Fiber surfaces to traits and feature level sets Jankowai & Hotz (2018a) (Section 3.7). A number of topology based methods have also been proposes such as Kuhn *et al.* (2017) and Doraiswamy *et al.* (2013). In Kuhn *et al.* (2017) the authors analyse pollutant clouds that emerge from volcanic eruptions. They make use of extremal graphs Kuhn *et al.* (2017) which are a sparse subsets of the Morse-Smale complex that encodes the spatial relationships of the extrema of the scalar field. The extremal graphs are tracked over time and used to detect major events of interest such as volcano eruptions and the formation of SO_2 clouds. Combining techniques from topology and computer vision Doraiswamy *et al.* (2013) introduces a framework for visualising the movement of cloud systems. They detect clouds as sublevel sets of infrared brightness temperature and precipitation over a region. Clouds of interest are picked out based on their persistence and tracked over time using an optical flow computation from computer vision. This is used to design a query system for cloud interactions and cloud tracking over various timescales.

Part II

Hypersweeps

Chapter 4

Data Parallel Hypersweeps for in Situ Topological Analysis

4.1 Introduction

Computational scientists use massive numerical simulations to study physical phenomena. As these simulations increase in size, techniques for analyzing and displaying the data are increasingly important. However, due to limited bandwidth to disk and in the human visual system, this increasingly depends on running analytics and visualization tools *in situ* during a simulation rather than *post hoc*.

The contour tree (see Section 3.4) can be annotated with geometric measures, such as volume and intensity, that are of significance to the science behind the data. In order to apply these tools at scale, recent work has built parallel algorithms and data structures for computing and using contour trees, first in data parallel environments, and in the future in hybrid clusters with on-node data parallelism. Data parallel algorithms to compute and augment the contour tree have been reported Carr *et al.* (2019); Carr *et al.* (2021b), but not the secondary computations such as geometric measures, branch decomposition, simplification and single isocontour extraction.

The first contribution of this part is to introduce data parallel algorithms for those secondary computations. To compute geometric measures we develop a method we call a hypersweep that is a modification of the parallel tree contraction algorithm Miller & Reif (1989). The hypersweep method arises naturally from the computation of the contour tree and unlike parallel tree contraction it respects the semantics of the contour tree as a data structure. For branch decomposition and simplification we replace the standard inherently serial priority queue computation with a local and trivially parallelisable algorithm. The second contribution is an implementation of those secondary measures in the open source VTK-m library. The final contribution is to link the resulting code with the existing *in situ* Cinema database to demonstrate viable data-parallel contour algorithms for the entire analysis and visualization pipeline.

We review the background literature in Section 4.2, then introduce the hypersweep in Section 4.3, showing how to adapt branch decomposition to data-parallel computation. Then we report our VTKm implementation and how we integrated it with the Cinema database in Section 4.4. Finally we evaluate the performance and show an application to the WarpX laser plasma particle accelerator simulation in Section 4.5.

4.2 Background

In this section we will expand the general background given in Part 3. We will add some more detail regarding parallel algorithms for computing contour trees and for tree operations. We will also discuss contour tree simplification in more detail.

4.2.1 Contour Tree Hyperstructure

Recall the Parallel peak prunning (PPP) algorithm from Section 3.4. The PPP algorithm batches superarc transfers from the merge trees to the contour tree, alternating between maxima and minima. In every stage leaves can be transfered in parallel because that is a local operation. To speed up computation long chains of degree two vertices are transfered in a single stage. Due to the specifics of the computation those chains need to be monotone in the function values at the vertices.

The original PPP algorithm has been recently extended to compute the augmented contour tree efficiently Carr *et al.* (2021b). The extension was to record the monotone chains from the merge phase to guarantee the ability to search for regular nodes in logarithmic time. The endpoints of the monotone chains are recorded as hyperarcs, similary to the already existing superarcs and regular arcs. The hyperarcs of the contour tree form what we call the hyperstructure Carr *et al.* (2021b).

We illustrate the idea of the hyperstructure with the right hand subfigure of Figure 4.3. The supernodes of the contour tree are labeled with the number of the iteration they are transfered in the merge phase of the algoritm. The supernodes in the tree are connected by superarcs and hyperarcs with small and large arc widths respectively. The hyperarcs store a monotone path of supernodes (sorted by value) that are collapsed in a single iteration of the merge phase.

We can think of hyperarcs as shortcuts for more efficient computation. Since the supernodes in a hyperarc are in monotone order we can insert regular nodes by comparing against the endpoints of the hyperarc. If the regular node's value is not in that interval, we move along the next hyperarc and skip a potentially large number of supernodes. Otherwise we use binary search on the supernodes in the hyperarc. In this paper we will describe how we can use the hyperstructure to speed up secondary computations such as geometric measures and contour extraction.

Since the merge phase of the contour tree algorithm and the hyperstructure process monotone paths an issue emerges with non-monotone paths. Non-monotone paths in the contour tree are referred to as W-structures Hristov & Carr (2019) because of the way they zig-zag up and down. We refer to the size of a w-structure as the number

of maximal monotone paths. W-structures are significant because they serialize the computation of the merge phase and can be used to show that persistent homology differs from branch decomposition Hristov & Carr (2019).

4.2.2 Simplification and Branch Decomposition

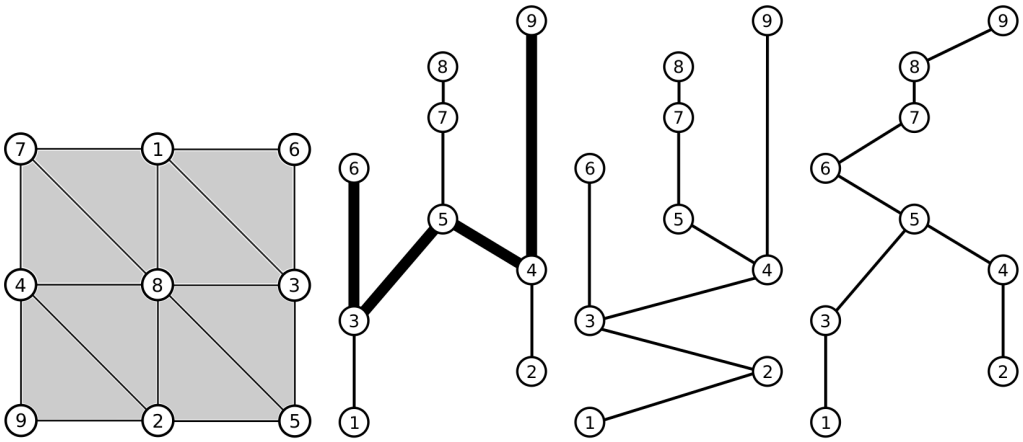
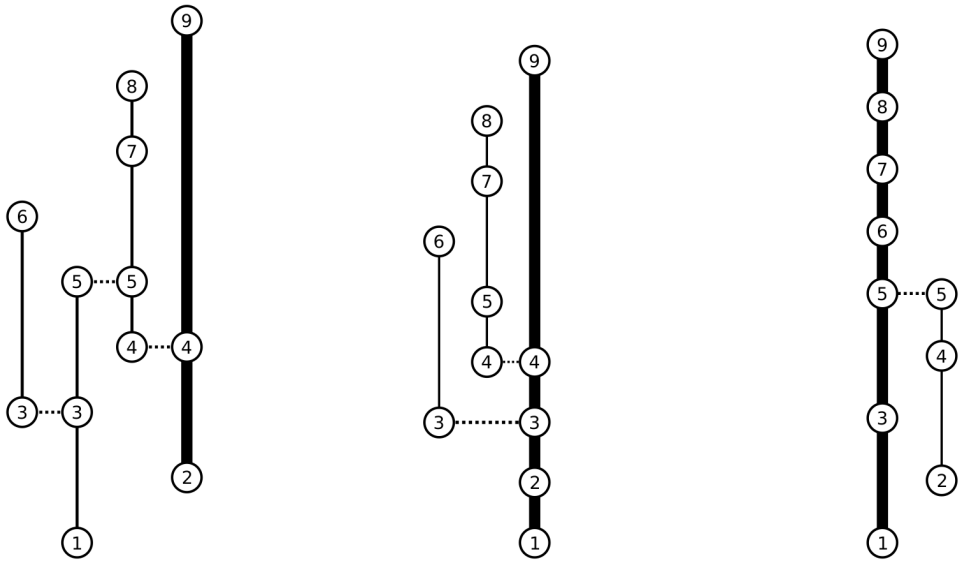


Figure 4.1: A simplicial mesh (a), that generates a contour tree (b) and the corresponding join (c) and split trees (d). The vertices are labeled with their height values. The thicker edges in the contour tree represent a W-structure Hristov & Carr (2019) which complicates the branch decomposition.

Once the contour tree has been computed, it can be simplified so that only significant features are represented. This is usually done by removing the least significant leaf edge in the contour tree, collapsing regular nodes if necessary, and iterating until only one master branch remains Pascucci *et al.* (2004a). Note that this is an inherently serial computation.

This simplification process forms a hierarchy of branches called the *branch decomposition*. For this purpose, "least significant" can be interpreted by computing the difference in function value between an extremum and a saddle, or by computing *geometric measures* Carr *et al.* (2010b) such as volume or integrated function value (hypervolume) for the set of contours corresponding to a given superarc or subtree.

A related idea is present in persistent homology Edelsbrunner *et al.* (2000), where the difference between a peak and a saddle in the sort order of the mesh vertices gives the *persistence*, and is used to pair, or cancel, peaks and saddles (or pits and saddles), or alternately, the difference in function value between peak and saddle. Recent work has confirmed Hristov & Carr (2019) however that the cancellation pairs from persistent



(a) Contour Tree (b) Join Tree (c) Split Tree

Figure 4.2: Branch decomposition of the contour and two merge trees from Figure 4.1 with the edges of the master branches in thicker lines. Each branch represent a feature in the data set.

homology are only guaranteed to match branches in the branch decomposition if no W-structures are present. In practical data, W-structures exist (as we will see in Section 4.5) and persistent homology gives a different result from branch decomposition. We use the term *height* of a feature to refer to the difference in value along a superarc to avoid confusion with the formal definition of persistence.

Given a simplified contour tree, visualization interfaces can be built that show only the most significant features or contours, and allow visual manipulation Carr *et al.* (2010b) of the remaining features, or extraction for subsequent processing with other algorithms. In essence, our goal is to replace the previous serial algorithms for geometric measure computation, simplification, branch decomposition and single isosurface extraction with data-parallel equivalents so that they can be run in an *in situ* environment efficiently.

4.2.3 Parallel Tree Operations

A fundamental parallel tree algorithm is parallel tree contraction Gibbons *et al.* (1994); Miller & Reif (1989). Parallel tree contraction is a bottom up technique where we start at the leaves of a rooted tree and move inwards in stages. In every stage all leaves with a different parent are processed independently in parallel. Once all leaves are processed

they are discarded (raked) and new vertices become leaves. If the tree is unbalanced the rake operation serializes the computation along chains of vertices of degree two. Those chains can be contracted using pointer doubling or a prefix scan. After a logarithmic number of rake and contract operations the whole tree is contracted to its root. At the end every vertex accumulates the value that corresponds to evaluating the expression over the subtree whose root is that vertex.

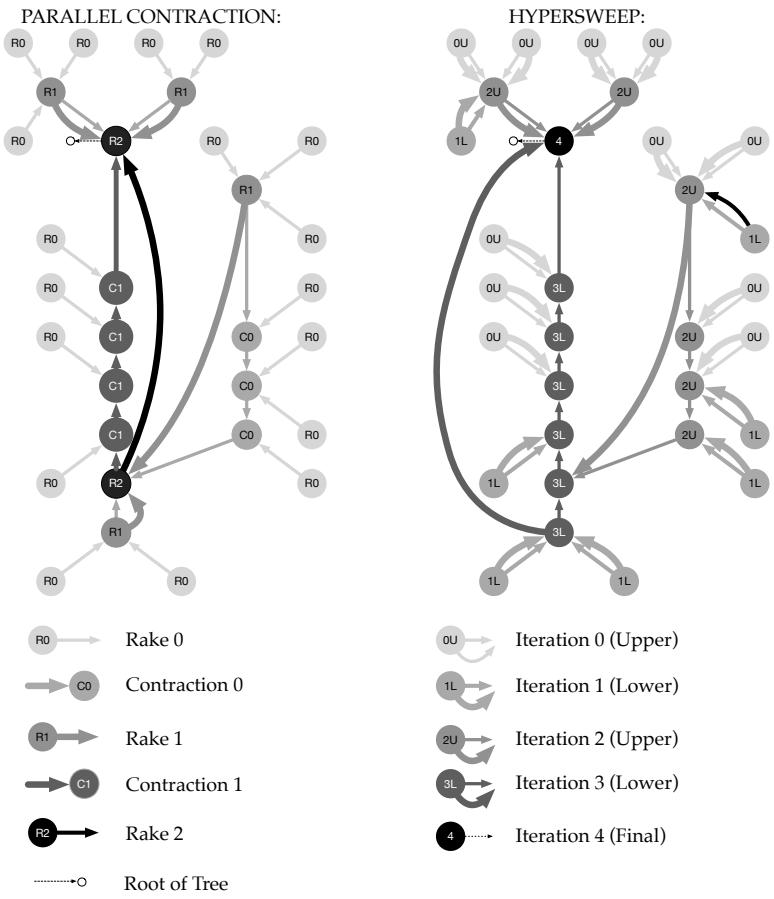


Figure 4.3: On the left is a contour tree whose vertices are annotated based on how and when they're processed by the parallel tree contraction algorithm. On the right is a hypersweep of the same contour tree annotated with the hyperstructure. While the two methods are similar, differences arise because PPP Carr *et al.* (2019) alternates upper and lower leaves, and because only monotone chains can be compressed.

As we have noted above, the merge phase of the PPP algorithm Carr *et al.* (2016) is a variation of parallel tree contraction, but with several differences. First, the hyperstructure only collapses chains whose vertices are monotone in value: this property is required to support binary search for data values along a path in the tree. Second, due

to the need to keep intermediate results updated, the hyperstructure transfers upper leaves and lower leaves in alternating passes. While it is tempting to view each pair of upper and lower iterations in the hyperstructure as equivalent to the contraction phases, variations are visible even in small trees, as shown in Figure 4.3.

4.2.4 The Cinema In Situ Database

Advances in processing power for extreme scale scientific computation have greatly outpaced data bandwidth and I/O, impeding visualization and analysis. The Cinema database Ahrens *et al.* (2014) is a large collection of images which are sampled based on time, visualization object and camera position, and stored along with metadata that allows interactive querying O'Leary *et al.* (2016). Cinema is used with image processing techniques to combine images to obtain new camera and time locations or even to reconstruct the original object using Depth Image Based Rendering Lukasczyk *et al.* (2018). Cinema has been implemented in ParaView as well as the open source Topology Toolkit TTK Tierny *et al.* (2017). However, since the images and the metadata are orders of magnitude smaller than simulation raw output they can be transfered for post hoc analysis and visualization. This requires sophisticated techniques for identifying features of interest, hence the interest in contour trees for analysis at scale. Our approach allows us to compute the triangles of connected components in situ and, by storing them as Cinema image collections, reduce their size and visualize large-scale simulation runs interactively on commodity hardware.

4.3 Hypersweeping Geometric Measures

In this section we will describe data-parallel computation of geometric measures such as volume and height (if not persistence), and to use them to construct branch decompositions. Geometric measures describe properties of a region bounded by a given contour, i.e. a region corresponding to a subtree of the original tree. For example volume is determined by all superarcs in the subtree, not just the final superarc at which the subtree is rooted. Hence we evaluate arithmetic expressions over subtrees of the contour tree and so we look to the parallel tree processing technique from Section 4.2.3.

Since parallel contraction is well-established, we will not illustrate the process in detail, restricting ourselves to the computations of interest, and commenting on how the variation of the hypersweep from parallel contraction affects the algorithmic analysis. The hypersweep algorithm as a variation on parallel tree contraction that follows the hyperstructure (see Figure 4.3 right) of the contour tree and we illustrate this in the left-hand column of Figure 4.4, where we compute an approximation of contour volume by counting the number of contained regular nodes Schneider *et al.* (2008b).

We know Carr *et al.* (2010b) that the number of regular nodes in a subtree approximates the volume of the regions represented by branches of the contour tree. While we could do a hypersweep with regular nodes rather than supernodes, it is less efficient.

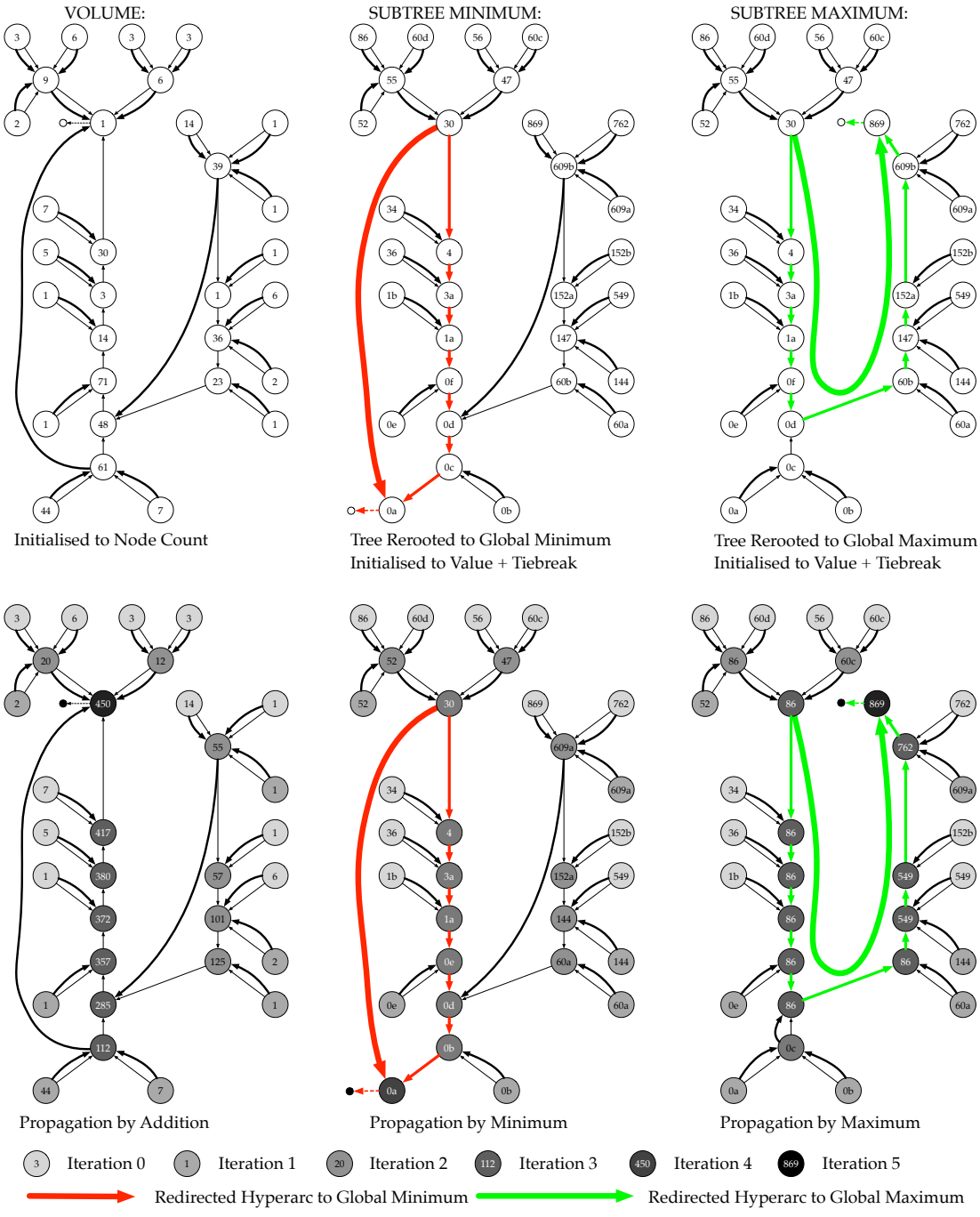


Figure 4.4: Hypersweep computation of geometric measures based on the parallel tree contraction Miller & Reif (1989). For volume approximation (Left), we initialize each supernode to the number of regular nodes on its superarc, then propagate towards the root with a prefix-sum. For sub-tree minimum and maximum (Centre and Right), we re-root the tree to the global minimum (maximum), initialize to the supernode's data value (using simulation of simplicity), then propagate by prefix minimum (maximum).

We therefore use prefix sum operations to compute the number of regular nodes on each superarc as the initial value at each supernode, as shown in the left column of Figure 4.4. We use shading to indicate the iteration in which these values are propagated inwards by prefix sums, resulting in the final tree sizes visible in the lower register.

In the absence of W-structures Hristov & Carr (2019), the chains in each pair of sweeps will remove the same supernodes as a single iteration of the parallel tree contraction: since this is a constant factor, the overall analysis is unchanged. In the presence of W-structures the hypersweep cannot be bounded by $O(\lg t)$ time complexity and $O(t \lg t)$ work where t is the tree size. In practice the work is still bounded by $O(t \lg t)$, and the time complexity is typically better than $O(\lg t)$ Carr *et al.* (2021b).

4.3.1 Branch Decomposition and Subtree Height

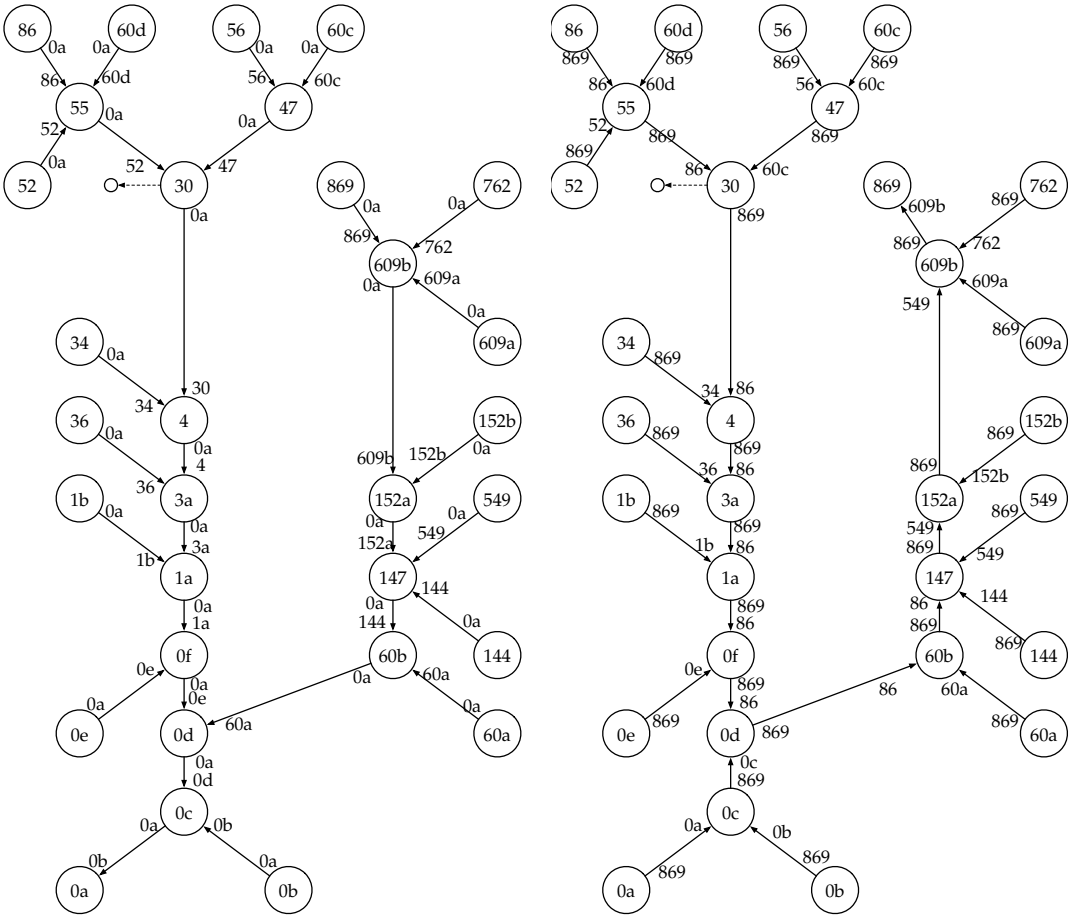
Once we have established subtree volume, we build branches by having each vertex choose locally the superarc with the highest ascent and descent. The branches are groups of adjacent superarcs that greedily maximize subtree volume or any geometric measure we have defined. We demonstrate this in Figure 4.6 with black dots on the edge adjacent to the best up and best down of each supernode. After each vertex chooses the "best" ascent and descent, we use pointer-doubling to collect the branches.

Building the standard branch decomposition Pascucci *et al.* (2004a) based on branch height is more difficult. When there are no W-structures in the contour tree each vertex can select the highest (or lowest) reachable maxima (or minima). In the presence of W-structures we need to compute the longest branch in every subtree. This makes the existing branch height decomposition difficult to compute in parallel. We will deal with this in more detail in Section 4.5.3. Now we will introduce an alternative geometric measure that is readily parallelizable.

Instead of branch height, we consider subtree height for the branch and all child branches. We define subtree height as the difference in function value between the maxima and the minima of a subtree. This means that we need the minimum and maximum values in every subtree from the root outwards. As we will see later Figure 4.11, this gives a slightly different branch decomposition than previous definitions, but only in the presence of W-structures.

We can now frame this in terms of a hypersweep operation: to find the minimum value in each subtree, we re-root the hyperstructure at the global minimum, then apply a hypersweep with the minimum operator. Re-rooting the hyperstructure is fairly straightforward: we select the global minimum m, and identify the hyperarcs along the path P between it and the previous root r, at a cost of at most $|P|$.

All paths from the leaves to the root terminate at the root r or at this path P. We convert this path to a new hyperarc (which may not be monotone) with at most the same number of iterations as before. This new hyperac is shown in the upper register of the middle column in Figure 4.4 as a thick red edge. We then hypersweep to propagate minima through the tree towards the minimum m, as shown. The right column of



Hyperstructure orients towards global minimum, so Value **outside** subtree is global minimum Value **inside** subtree comes from minimum tree

(a) Minimum subtree values.

Hyperstructure orients towards global maximum, so Value **outside** subtree is global maximum Value **inside** subtree comes from maximum tree

(b) Maximum subtree values.

Figure 4.5: The minimum and maximum subtree values computed using the hypersweep values from Figure 4.4.

Figure 4.4 shows the re-rooting and hypersweep to compute subtree maxima.

In the next stage of the computation, shown in Figure 4.5, we annotate every edge in the tree with *two* values: the minimum in the direction of the hypersweep, and the minimum in the other direction. Of these, the minimum in the hypersweep direction is set to the value just computed. The minimum in the other direction will always be the global minimum, since it is the new root of the tree.

For example, in the left top corner, the vertex with value 86 forms a subtree, and the propagated minimum value, 86, is the value we use when pruning towards the root: the global minimum value, 0, is the value when pruning away from the root. Now, for each possible pruning (i.e. at each end of the superarc), we add the value of the supernode itself, then take the maximum and minimum of the three values: thus, if the supernode value is the lowest, it replaces the minimum, if the highest, it replaces the maximum. Finally, we subtract minimum from maximum to get the subtree range.

Considering vertex 86 once more, pruning at the lower end of superarc $86-55$ gives a subtree minimum of 86 and maximum of 86. We substitute 55 for the minimum, and compute a subtree height of 31. Further in, at the lower end of superarc $30-4$, the maximum is 86 in the upwards subtree and the minimum 30. Replacing 30 with 4, we compute an upwards subtree height of 82.

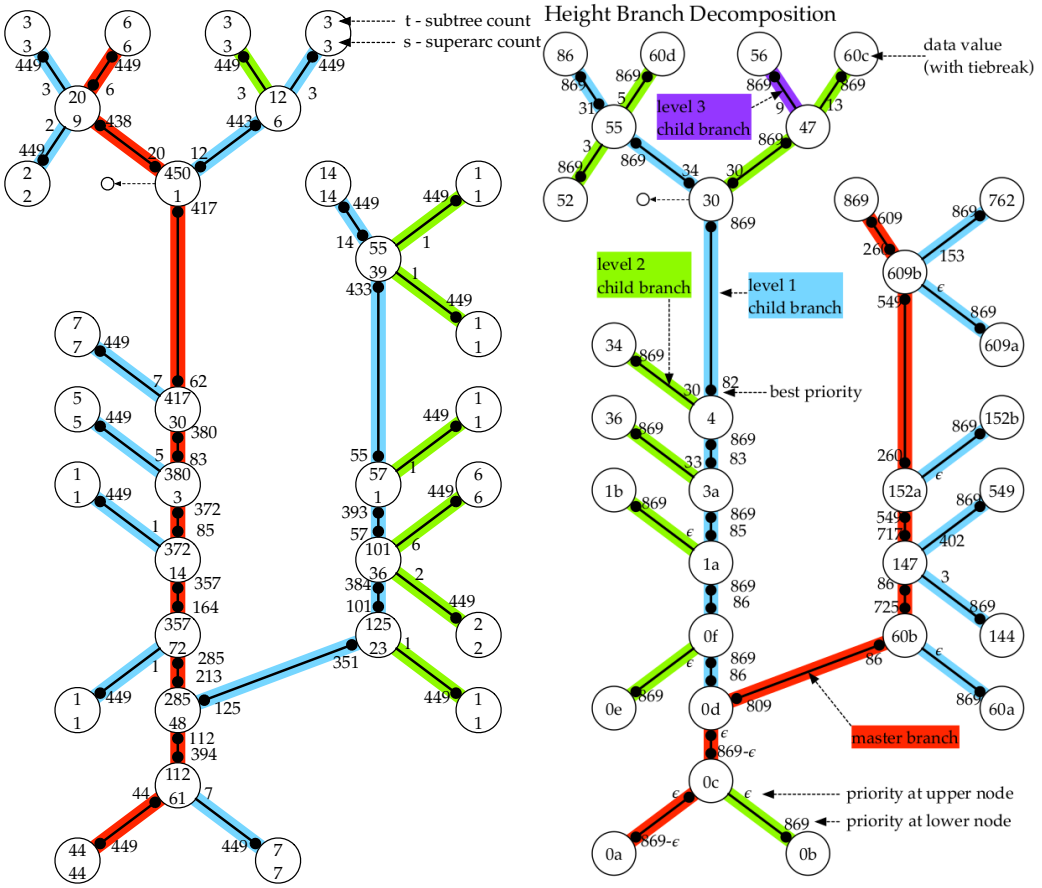
4.3.2 Simplification

Having computed our geometric measures and branch decomposition, simplifying to a threshold amounts to ignoring branches of the contour tree that fail a logical test. For example, suppose we want to ignore all branches that involve less than 1% of the data. This is achieved by testing all superarcs to see whether their volume (or height) is over the threshold, which is trivial to do in parallel. If desired, the "weight" of the pruned branch can be retained by keeping the terminal superarc as an augmenting node in the simplified tree.

4.3.3 Feature Extraction

Once the contour tree has been computed, decomposed and simplified, visualization interfaces extract contours corresponding to selected superarcs. In prior work Carr *et al.* (2010b), the user interactively selected contours and manipulated them visually. While this is still possible with the data-parallel contour tree, one goal of *in situ* visualization is to defer user interaction until later. We adopt an alternate solution - local contours Carr *et al.* (2010b), where we choose a relative isovalue on each branch - normally 50%, or halfway along it.

Previous work Carr *et al.* (2010b) adapted the continuation method Wyvill *et al.* (1986) to extract single contours, but this approach is essentially serial. Instead we extract a contour for a branch using marching cubes and a method based on searching the contour tree Weber *et al.* (2007a). First we use a parallel implementation of marching cubes to extract an isosurface for the isovalue of that branch. Next we filter out the triangles produced by marching cubes that do not belong to the branch.



(a) Volume branch decomposition.

(b) Height branch decomposition.

Figure 4.6: The branch decomposition of the same contour tree using the geometric measures of volume (a) and height (b). Notice that the two geometric measure produce two slightly different branch decompositions. The master branch (in red) in the two decompositions is different.

To determine if a triangle belongs to a branch consider a mesh edge u, v that intersects the triangle. Since the path from u to v in the mesh is monotone there is monotone path from u to v in the contour tree. Therefore along that path there is a superarc whose endpoints' values contain the isovalue for the branch. We search for that superarc with the hyperstructure because it supports efficient search for regular points at logarithmic cost Carr *et al.* (2021b). If that superarc belongs to the branch we keep the triangle, otherwise we discard it.

Each such contour can be extracted in $O(k \lg T)$ time, where k is the size of the entire isosurface, and $O(\lg T)$ is the cost of searching the hyperstructure to find the corresponding superarcs. For a small number of contours (e.g. 10 or 20), we iterate over their superarcs and values to generate them, with the advantage that we will extract them as separate surfaces and can render them accordingly. For large numbers of contours, each mesh cell (or mesh edge) can search for the corresponding path(s) in the contour tree and compare them all at once, but we have not yet implemented this variation.

4.4 Implementation

To demonstrate integrating our parallel methods into a full visualization pipeline, we developed the "contour visualizer" application prototype. Our goals in developing this application were: (i) to extract a representative set of contours from the scalar field with minimal user interaction; (ii) utilize high-performance computing to handle large-scale data sets; (iii) to use standard scientific visualization libraries for easy integration into existing project.

We implemented the hypersweeps described in Section 4.3 as part of the VTK-m project, and integrated them with the existing Cinema database application, using a two stage visualization pipeline. In the first stage, we extract, compress and store features from scalar fields in a Cinema database. In the second stage, we read images from this database, reconstruct features from depth images and visualize them. All of the methods developed have been contributed to the development branch of VTK-m, and are available for use.

Our input is assumed to be a standard VTK image format. While our current implementation works with regular, rectilinear grids, the underlying algorithms employ the topology graph abstraction referred to in Section 4.2, and are valid for any simply connected mesh, subject to writing suitable adaptor classes.

We compute the contour tree of the scalar field, assuming marching cubes connectivity, using the VTK-m Moreland *et al.* (2016) implementation of the parallel peak pruning algorithm Carr *et al.* (2019); Carr *et al.* (2021b). Subsequently, we compute the branch decomposition (as described in Section 4.3) either using subtree height or volume as the simplification measure and simplify the branch decomposition to a specified number of branches.

As described in Subsection 4.3.3, we then simplify the contour tree to the top 10 most important branches, and extract one representative contour per branch in local

contour mode. At present, we usually choose the 50% isovalue on each branch, but we have also used the 1% isovalue to select contours very close to the critical point: in future we expect to choose multiple contours along each branch.

After single contour extraction *in situ*, the first stage is complete, and we save depth images from varying camera positions for later reconstruction Lukasczyk *et al.* (2018) based on a TTK Tierny *et al.* (2017) implementation in order to avoid saving large meshes of millions of triangles.

The second stage supports post hoc exploration of the data artifacts stored in the Cinema database. We read all depth images in the cinema database and reconstruct each feature individually using the TTK Tierny *et al.* (2017) implementation of the VOIDGA algorithm Lukasczyk *et al.* (2018). The quality of the reconstruction depends on image resolution, camera placement and number of viewpoints in the Cinema database.

As with the Cinema database in general, our project can use different front ends. For some purposes, we use ParaView and TTK, but for others, we implemented a simple web server and web interface to reduce the learning curve for end users. We implemented this using node.js for the server and Three.js for the front end.

The quality of the reconstruction can vary but does not need to be perfect, only good enough for the user to get a general idea of the data. Should the user require the original features they can be retrieved at a higher bandwidth and time cost, and we will explore the best parameter choices for *in situ* visualization in the future.

This visualization pipeline is an improvement upon previous ones such as Biedert & Garth (2015). Every step in our pipeline is fully data-parallel and it is implemented using popular open source visualization libraries such as VTK-m and TTK. Furthermore our pipeline adds the additional step of reconstructing the depth images in 3D.

4.5 Evaluation

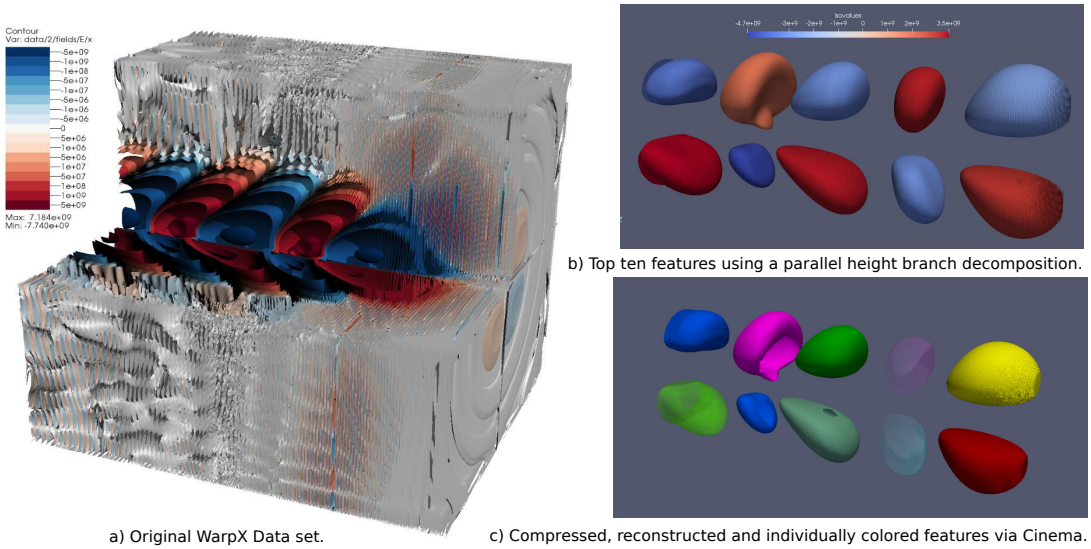
In this section we will discuss the evaluation of our application. We will consider some example data sets, discuss the quality of contour selection and present performance timings.

4.5.1 Application Example - WarpX

Figure 4.7 shows the application of the automatic contour selection to the transverse electric field (Ex) of a WarpX laser plasma particle accelerator simulation. Plasma-based accelerators use short ($\leq 100fs$) ultrahigh intensity ($\geq 10^{18}W/cm^2$) laser pulses to drive waves in a plasma. Electrons that become trapped in the plasma (or externally injected electron or positron beams) are then—much like a surfer riding a wave—accelerated by the wave to high energy levels. Understanding the structure of the electric forces generated by the plasma wave is critical to the design and optimization of plasma-based particle accelerators and understanding of the fundamental physical phenomena. In this context, the difference in function value (i.e., height of arcs in the

contour tree) is an important measure of the strength of the electric forces generated by the corresponding feature (i.e., contour) in the electric field.

As Figure 4.7 b) shows, using height as an importance metric allows us to automatically identify the features with the largest focusing gradients in the transverse electric field Ex, describing the primary structures of the electric field generated by the plasma wave driving particle acceleration. By rendering the contours in situ and storing for each contour a separate depth-image in a Cinema database, users can interactively explore, visualize, and compose the features post-hoc. By storing the additional metrics computed from the contour tree (e.g., volume and persistence of contours) alongside the generated depth images, enables quantitative analysis of the contour-based features and interactive query of the Cinema database to search for relevant contours.



a) Original WarpX Data set.

b) Top ten features using a parallel height branch decomposition.

c) Compressed, reconstructed and individually colored features via Cinema.

Figure 4.7: a) Isosurface visualization of the transverse electric field Ex of a WarpX laser plasma particle accelerator simulation Carr *et al.* (2019). b) Visualization of the ten most-significant contours detected automatically using a branch decomposition of the contour tree using our data-parallel, height-based simplification method that correctly captures the topology of the data set. c) For interactive, post-hoc visualization, we compute and store features in a Cinema image database in situ and reconstruct them via a web interface. We store features individually and this allows us to manipulate their properties such as color, scale, opacity, etc.

Next we evaluate the compute performance of our implementation (Sec. 4.5.2) and how well it picks out significant contours (Sec. 4.5.3).

Table 4.1: The dimensions of our test data sets and timings for contour tree computation. Note that the number of supernodes and timings for all data sets differ from the ones reported in Carr *et al.* (2021b) because we are using marching cubes connectivity.

Dataset	**Dimensions**	**Contour Tree Supernodes**	**Compute Tree seconds**
Hydrogen Atom	128x128x128	13,038	0.399
Aneurism	256x256x256	65,625	2.793
Bonsai	256x256x256	192,067	3.153
WarpX_E_x	6791x371x371	288,807	317.191
Asteroid	500x500x500	881,831	23.160
Backpack	512x512x373	7,441,922	27.990
Spathorhynchus	1024x1024x750	44,554,912	330.926
Kingsnake	1024x1024x795	55,778,125	268.833
Pawpawsaurus	958x646x1088	89,117,386	352.491
GTopo30 at 0.03125	675x1350	72,276	0.236
GTopo30 at 0.0625	1350x2700	271,772	0.735
GTopo30 at 0.125	2700x5400	991,480	2.571
GTopo30 at 0.25	5400x10400	3,579,117	10.387
GTopo30 at 0.5	10800x21600	12,688,670	44.054
GTopo30 at 1.0	21601x43201	36,912,523	172.301

Table 4.2: Once the contour tree and hyperstructure have been computed, hypersweeps to compute secondary properties are highly efficient, adding less than 1% extra time. Our modified branch decomposition, which uses multiple hypersweeps, is a negligible additional cost. The last two columns present the ratio of the time it takes to compute the hypersweep (HS) to the time to compute whole contour tree (CT) and the time it takes to compute the branch decomposition (BD) to the time to compute the contour tree (CT).

Dataset	**Hypersweep seconds**	**Branch Decomp seconds**	**Ratio HS / CT**	**Ratio BD / CT**
Hydrogen Atom	0.001	0.025	0.33%	6.47%
Aneurism	0.003	0.039	0.12%	1.39%
Bonsai	0.007	0.072	0.23%	2.30%
WarpX_E_x	0.005	0.055	0.01%	0.01%
Asteroid	0.018	0.258	0.08%	1.11%
Backpack	0.118	1.431	0.42%	5.11%
Spathorhynchus	0.459	7.299	0.13%	2.20%
Kingsnake	0.589	8.887	0.21%	3.30%
Pawpawsaurus	0.979	13.841	0.27%	3.92%
GTopo30 at 0.03125	0.002	0.014	0.98%	6.21%
GTopo30 at 0.0625	0.004	0.036	0.65%	4.95%
GTopo30 at 0.125	0.004	0.036	0.18%	1.41%
GTopo30 at 0.25	0.012	0.108	0.12%	1.04%
GTopo30 at 0.5	0.038	0.353	0.08%	0.80%
GTopo30 at 1.0	0.381	3.981	0.22%	2.31%

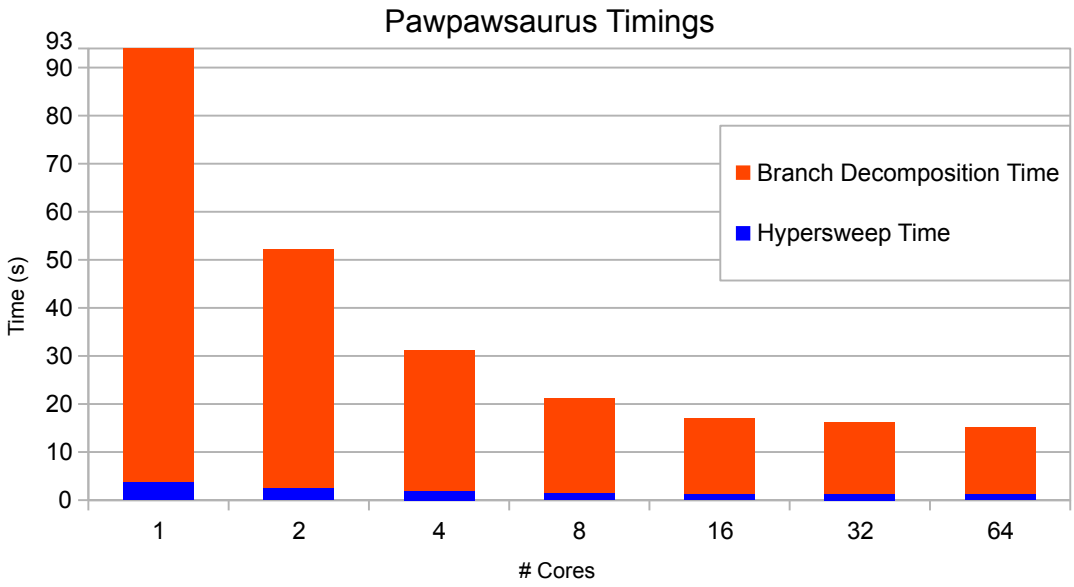


Figure 4.8: Subtree height branch decomposition (red) and hypersweep (blue) on Pawpawsaurus. While the scaling plateaus after 8 cores, the performance is overall good especially compared to contour tree computation (see Table 1).

4.5.2 Performance

As noted above, our implementation is freely available in the open source VTK-m library Moreland *et al.* (2016). However there is no implementation of branch decomposition in any other actively maintained visualization library (TTK and VTK). To ensure consistency between methods, we re-implemented branch decomposition in serial. It performed with about the same running time as the parallel branch decomposition on a single core. We have not included those specific running times because our serial branch decomposition was implemented as reference for comparison not with performance in mind.

We ran tests on standard data sets well known in the visualization community or that we have used previously Carr *et al.* (2019); Carr *et al.* (2021b), and refer the reader to the appendices of those papers for full details. The Asteroid dataset is freely available courtesy of LANL, the WarpX dataset is not freely available at present.

Our primary test system is the NERSC Cori supercomputer at Lawrence Berkeley National Laboratory, whose Haswell compute nodes have two 16-core Intel® Xeon ™ E5-2698 v3 CPUs with two hyperthreads per core, clocked at 2.3 GHz and with 128GB DDR4 main memory at 2133Mhz. We compiled and used the VTK-m library with Intel's Thread Building Blocks (TBB) threading API.

We first computed the augmented contour tree for each data set using VTK-m's contour tree filter Carr *et al.* (2021b). Next we compute the branch decomposition of

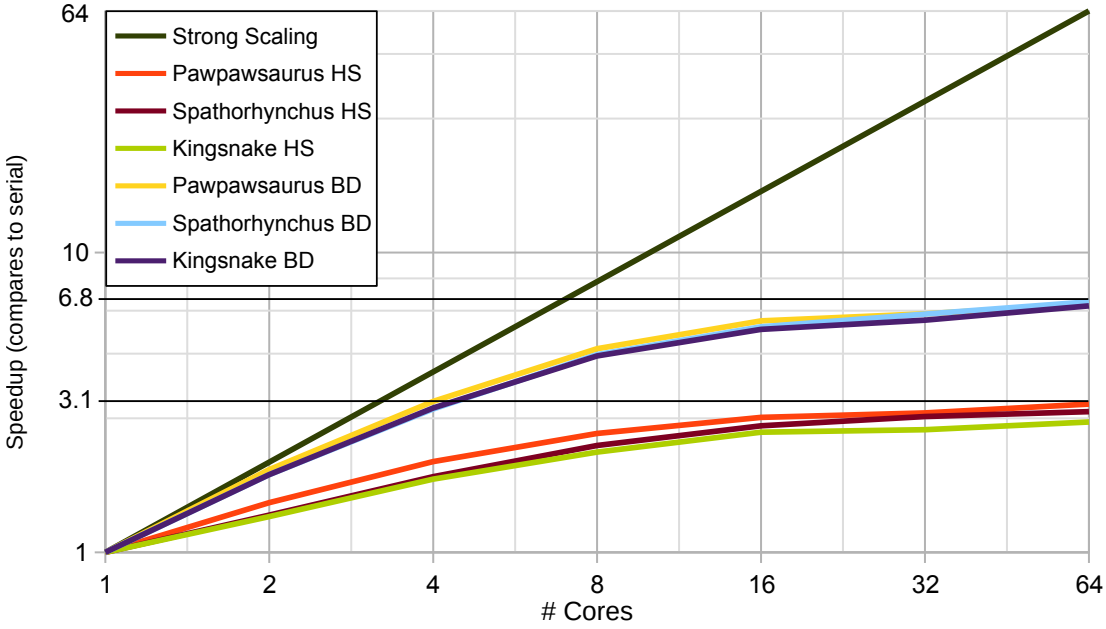


Figure 4.9: Scaling of 3D data for up to 64 cores and TBB on Haswell (log/log). The black line shows the ideal strong scaling. Hypersweep (HS) and branch decomposition (BD) are related and have similar scaling patterns: it is possible that the cause is external (VTK-m).

every data set with a range of 1, 2, 4, 8, 16, 32 and 64 cores. Finally we compute the branch decomposition over the GTOPO30 data set with 64 cores, but with different scales of the data. This way we can study scaling on a set of related data sets.

In Figure 4.8, we show timings for the Pawpawsaurus data set. We have chosen Pawpawsaurus because it is one of the largest data sets we have available in terms of regular and super node count. We therefore expect to see the scaling of the hypersweeps, rather than the cost of initialising parallel data structures. Here, we see the most performance gain in going from 1 to 2 cores and then to 4 cores and 8 cores. This is also visible in Figure 4.9 where the speedup of the hyperswep and the branch decomposition is 3.1 and 6.8 respectively.

Similarly Figure 4.10 suggests that while the scaling is not linear (gray area in the plot) the performance is still good in practice. This is further supported by Table 4.2 where we can clearly see that the hypersweep and branch decomposition are only a small fraction of the computation time of the contour tree. On average the branch decomposition is only 1.76% of the contour tree computation time, so we do not yet see the need for further optimization.

An important reason for the good practical performance of our methods is the topological complexity of the data sets. Remember that our methods do not scale with size of the input mesh, but rather the number of supernodes of the contour tree of the

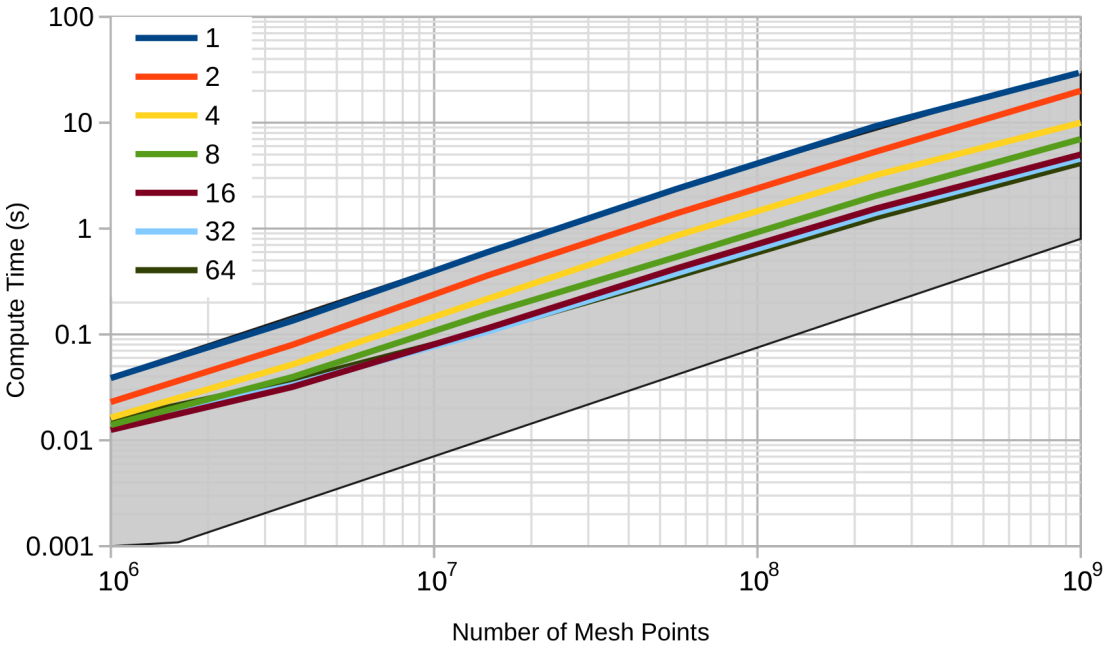


Figure 4.10: Scaling using 1 to 64 threads on the 2D Scaled GTOPO Datasets (log/log). The grayed out polygon is perfect weak scaling.

mesh, As we can see in Table 4.2 the number of supernodes in most tests is roughly an order of magnitude smaller than the number of regular nodes. Even though a serial method would have sufficed in some tests the need for parallelization will become even more apparent in the future with data sets with more topological complexity or sampling noise.

Furthermore (as pointed out by a reviewer) there are many practical situations, such as time varying domains or ensemble runs, where multiple contour trees need to be computed. For each contour tree we may need multiple branch decompositions if we do not know which geometric measure would be most useful beforehand. Those computations add up and any speedup over a serial implementation with optimal work complexity is valuable. Finally when accelerator devices such as GPUs are used for contour tree computation our parallel implementation allows us to avoid the high cost of inter-device data transfers to CPU for secondary computations.

4.5.3 Feature Significance

In this section we will consider how the difference between our subtree height decomposition and the standard branch height decomposition impacts feature selection. In Table 4.3 the two branch decompositions typically differ in only a small number of branches. Moreover, we know from Section 4.3 that the two are identical in contour trees with no W-structures (i.e. those with W diameter of 2 or less. In the table,

Dataset	Branches	W Diam	Difference	
shockwave	333	3	0	0.0000%
marschner_lobb	810	4	0	0.0000%
neghip	976	4	0	0.0000%
hydrogen_atom	6,532	4	0	0.0000%
aneurism	33,139	4	0	0.0000%
bonsai	96,993	5	8	0.0082%
tooth	151,302	5	4	0.0026%
statue_leg	223,469	6	13	0.0058%
foot	444,616	7	44	0.0099%
mri_ventricles	1,159,963	6	77	0.0066%
skull	1,130,490	7	155	0.0137%
backpack	3,813,085	7	315	0.0098%

Table 4.3: Differences from the standard branch decomposition Pascucci *et al.* (2004a). Both decompositions have the same number of branches, but some leaves can be paired differently. This is due to differences between branch height and persistence in the presence of W-structures Hristov & Carr (2019).

we see that this is the case, and that in fact, the smallest W-diameter where different decompositions emerge is 5.

In some data sets, the standard branch decomposition is less effective than our new parallel-friendly subtree height decomposition. In Figure 4.11 we show the result of choosing the top 20 features with the two methods. A large boxy object is visible when subtree height is used, but not when branch height is used. The relevant structures in the contour tree are the six illustrated branches (out of over $3,000,000$ total branches). Notice that the W-structure rooted at $0b$ means that the standard branch decomposition treats this feature as less important, but the new subtree height decomposition, which looks at the height of the entire subtree, displays it.

This does not indicate that the branch height decomposition is invalid, merely that it is imperfect, and that the subtree height is similar and similarly imperfect. However, the new height decomposition is easier to compute in parallel, which is worth having.

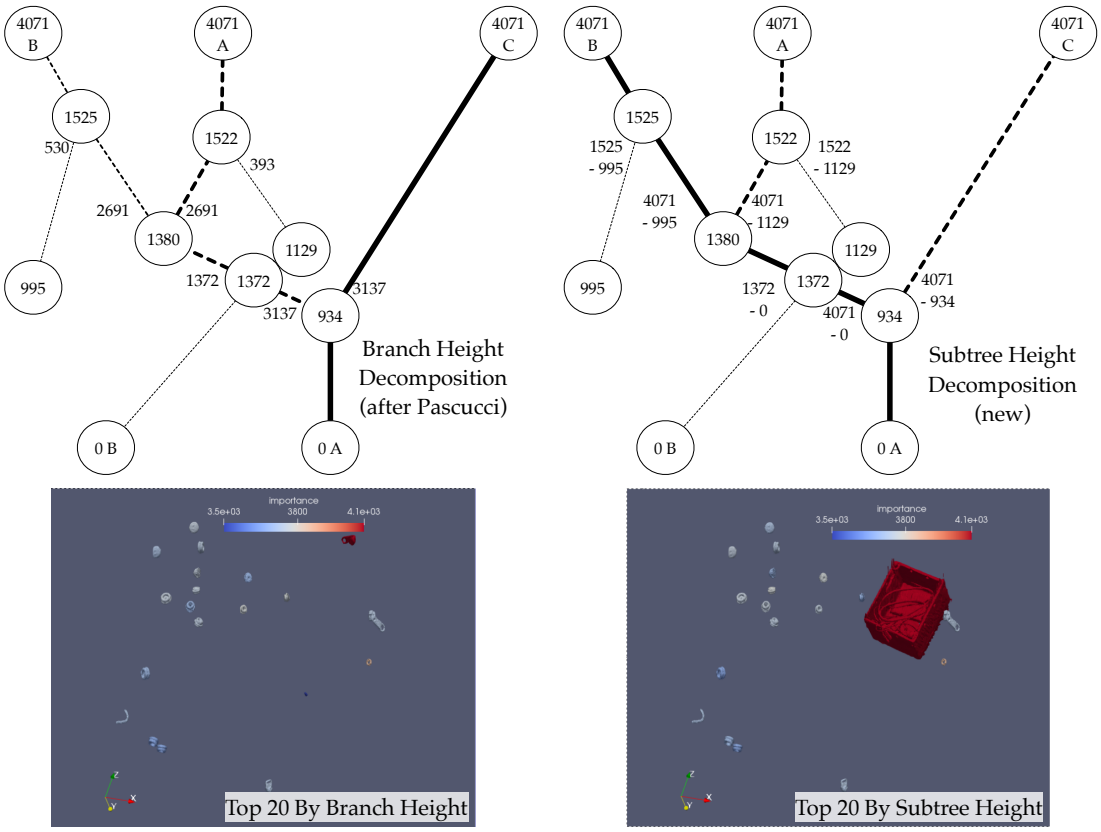


Figure 4.11: W-structures in the Backpack data set. Because of a W-structure ending in 0*b*, the left subtree at 934 has a larger overall height than the right subtree, giving a different branch decomposition than Pascucci's Pascucci *et al.* (2004a). On the right: the top 20 features chosen with each method. While the standard branch decomposition detects the box as feature 39 in order of significance. The subtree height decomposition works better than the persistence based decomposition in this instance.

Part III

Convective Clouds

Chapter 5

Cartesian Fiber Surfaces for Trivariate Visualisation

5.1 Introduction

Convective clouds play a vital role in regulating our weather and climate, but their formation and transport is not fully understood. Convection is governed by the dynamics and thermodynamics of the atmosphere which are based on interactions of temperature, vertical velocity and moisture. The exact relationship between these variables and convective cloud formation is yet unknown. The lack of an accurate physical representation of clouds reduces the reliability of numerical atmospheric simulations and impacts both short term weather prediction for flight coordination and agriculture as well as long term prediction of human activity on climate.

In this part, we describe a tool developed in collaboration with atmospheric scientists to study convective cloud formation in and above the *planetary boundary layer* of the atmosphere. At present, a common technique employed by atmospheric scientists is to simulate a surface-released decaying passive tracer in numerical atmospheric simulations Couvreux *et al.* (2010). This tracer enables study of the bulk movement of air without affecting the simulation (passive) or building up over time (decaying). However this has not yet led to a fundamental understanding of the relationship between the underlying physical variables such as water content, temperature, and vertical velocity in convective cloud formation.

In this part we present an application for trivariate visualization that takes as input three scalar fields defined on a common three dimensional regular grid. We define the main features as the volumes contained in an isosurface of a scalar field derived from the concentration of the passive tracer gas in the system. This identifies spatially where convective clouds are triggered. The next step is to study the physical properties (temperature, humidity, buoyancy) of those regions. We enable this by projecting those volumes onto the continuous scatter plot of pairs of those physical properties.

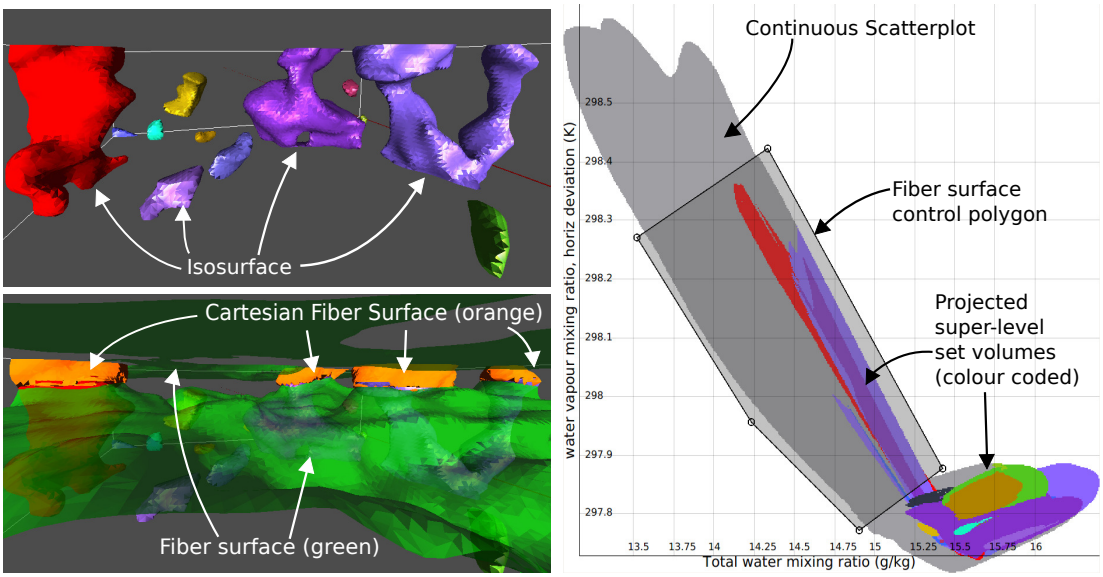


Figure 5.1: Top Left: an isosurface of a derived scalar field tracking air movement, color-coded to indicate the connected components of the super-level sets. Right: the scatterplot shows the probability distribution function of two other variables. The individual connected components of the super-level set (contained in the isosurface) are projected onto it, showing how regions on the top left map to the variables on the right. Bottom Left: fiber surfaces can then be defined via a control polygon and compared to the isosurfaces to examine the properties of parts of the objects. In this example we've defined a control polygon over an area in the scatterplot where data points have low moisture and high temperature. The Cartesian fiber surface is the boundary of the intersection of the volume contained in the fiber surfaces and the isosurface.

This facilitates hypothesis formation about the relationship between the three scalar fields by comparing the shape and distribution of features and their projections.

The application allows the user to brush areas of interest in the projections of the features and inversely map back to a spatial representation via fiber surfaces. This refines scientists' understanding of the phenomenon in two key ways. Firstly, the fiber surface includes all other areas in the domain (not in the features) which have similar properties. This has allowed scientists to identify areas which inhibit convective cloud formation. Secondly the intersection of the volume contained in the fiber surface and the volume contained in the isosurface allows the user to refine their features and study their inner structure. To avoid costly geometric intersection tests we use a novel technique for isosurface and fiber surface contained volume intersection called Cartesian fiber surfaces.

Using this tool, domain scientists were able to formulate mechanistic hypotheses on how the physical variables, describing temperature, moisture content and vertical velocity combine to trigger convective clouds. Specifically, cloud-triggering structures appear to be embedded in the nodal intersections of horizontal segments of rising moist air and away from surfaces of low environmental density aloft (which form barriers to the rising structures). And second, cloud-triggering structures appear to be constituted by concentric regions of increasingly cool and moist air.

We begin our discussion with background on convective cloud formation in Section 5.2. We describe our approach for volume intersection called Cartesian fiber surfaces in Section 5.3 and discuss the implementation of the application in Section 5.4. Section 5.5 then gives a case study of how this application supported reasoning in atmosphere science, and the insights derived through its use.

5.2 Convective Clouds in Earth's atmosphere

Convective clouds form when rising air with adequate moisture cools below the saturation point (dew-point temperature). The excess moisture then condenses into droplets releasing latent heat of condensation that increases the buoyancy and causes the air to rise further, in turn leading to further cooling and condensation. The formation of convective clouds thus requires *triggering* through initial condensation and then becomes self-sustaining.

The buoyant and moist parcels causing the onset of convection primarily originate from the Earth's surface where they are created through fluxes of moisture and heat by processes occurring at the surface (plant respiration, evaporation, radiative heating, etc). To represent the triggering of convection in contemporary weather prediction and climate models (which have inadequate spatial and temporal resolution to explicitly represent the formation of convection) it is necessary to formulate simplified mathematical models which are able to predict when clouds form. This in turn necessitates a physical model of the coherent structures that trigger clouds. The study of the formation of convective clouds therefore amounts to quantifying the transport of moisture and heat from the Earth's *boundary layer* into the *cloud layer*, and identifying exactly which air-parcels in the sub-cloud layer trigger the formation of clouds.

Identification and (quantification) of cloud-triggering coherent boundary structures is complicated by the fact that: 1) vertical transport is carried out by more besides these coherent structures and 2) both moisture and heat contribute to the buoyancy which make air-parcels rise. Specifically, vertical transport of moisture and heat (internal energy) in the boundary layer can be decomposed into two contributions. First, the local or down-gradient turbulent transport by small eddies (flow opposing the main current) causes diffusion into regions of lower concentration. Second, non-local transport by individual coherent structures (localised air parcels) carries larger perturbations in moisture and heat, and provides the impetus for triggering convective clouds.

Although the first diffusive processes for local transport can easily be modelled, (as a function of the mean vertical gradient) a firm physical understanding of the properties

of the coherent boundary layer structures providing non-local transport is still missing, limiting our ability to model evolution and transport by these structures. By enabling analysis where the spatial structures of scalar co-variances inside and outside of cloud-triggering coherent structures can be identified, the tool presented in this work enables the formulation of a conceptual physical model of convective cloud triggering. This in turn will enable better representation of convective triggering in weather and climate models.

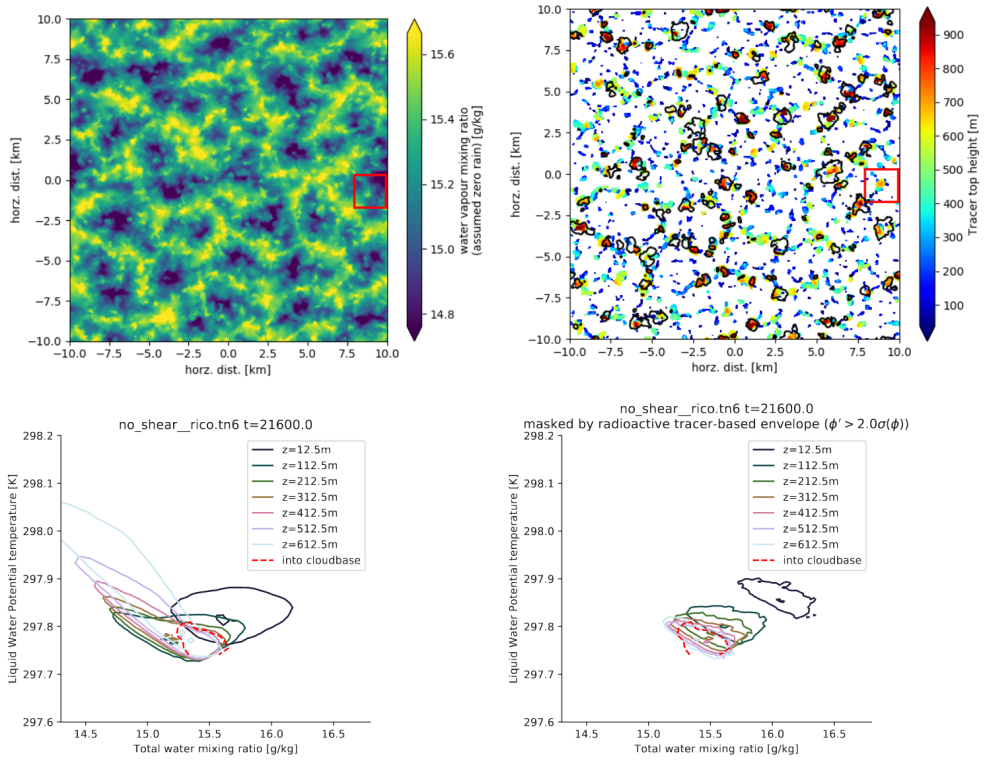


Figure 5.2: State of the art visualizations of cloud convection triggering Denby *et al.* (2022). Top left: horizontal cross-section of water-vapour concentration at height $z =$ 300m. Top right: Maximum altitude at each horizontal location where the tracer field exceeds a threshold value with cloud condensate regions (black outlines). The marked region contains a boundary layer structure about to trigger a cloud. Bottom: Contour lines of water vapour and temperature by altitude in the data domain, with air entering clouds at cloudbase (red dotted line). Bottom left: the entire data set with isosurfaces of the tracer. Bottom right: The same data set restricted to an isosurface of the passive decaying tracer, showing a linear development and suggesting that isosurfaces identify structures triggering convection.

5.2.1 Domain Science Related Work

One of the current techniques for tracking non-local transport is a decaying passive tracer (as in Couvreux *et al.* (2010)) released from the surface to track how air aggregates horizontally and rises vertically. Regions where there is a high-concentration of the passive tracer (compared to the variability in the horizontal plane) thus indicating that the airmass originated from the surface. By using the decaying tracer to define the coherent structures we avoid *a priori* assumptions about the values of physical state variables (temperature, vertical velocity, moisture content, etc) within coherent structures. This method has been shown Denby *et al.* (2019) to track air with properties necessary to trigger convective clouds, further supporting its use to define coherent boundary layer structures. However, this method relies on a tracer which is passively transported with the flow and does not use the physical properties of air to predict which air masses will trigger clouds, and so cannot in itself be used to produce a physical model of the triggering of clouds.

Previously, these structures have been studied in numerous ways, all characterised by the fact that they principally visualise at maximum two spatial dimensions of the underlying data, and when viewing more than one scalar field often trade the number of spatial dimensions visualised with the number of scalar fields (see for example Denby *et al.* (2019)). Specifically, 2D cross-sections and aggregations along a single dimension in 3D (to create a 2D field) are often used, together with joint distributions of the scalar values as a function of height (visualised by density contours), and finally once identified the structures are studied by measuring their shape, orientation and size, and plotting distributions of these.

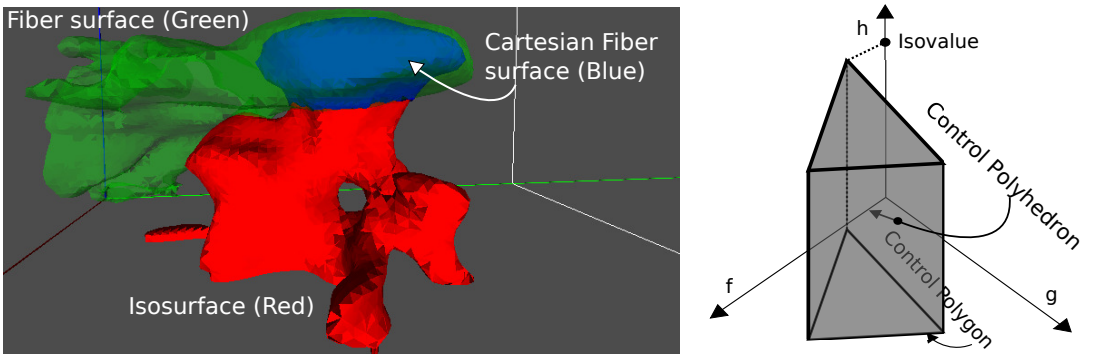
Figure 5.2 shows examples of visualisations commonly employed. Specifically, Figure 5.2 (top left) shows a horizontal cross-section of water vapour concentration below cloud and (right) the vertical height at which the tracer concentration exceeds a height-dependent threshold (the threshold corrects for the vertical gradient and the horizontal variability in the tracer concentration, see Couvreux *et al.* (2010) for details). These subfigures firstly exemplify the challenge in defining cloud-triggering structures using the physical fields themselves, given the diffuse distribution of water vapour and secondly serve to demonstrate how clouds and boundary layer structures are typically co-located (this visualisation was used to select subdomains to study as will be detailed in Section 5.5).

As an example from Denby *et al.* (2019) Figure 5.2 (bottom) displays the joint distributions of water vapour concentration and potential temperature (absolute temperature corrected for cooling due to adiabatic expansion) at increasing heights in the boundary layer with and without the use of the passive tracer to pick out coherent rising structures. For each height two contours are plotted, the innermost contains 5% highest density of points and the outermost the 95% points. Using these contours it is thus possible to visualize the mean and spread of the joint distribution and to compare them at different heights. As well as the distributions in the boundary layer, the distribution of air entering individual recently formed clouds at cloud-base is shown in red. Comparing the subfigures, it is clear that by using the decaying tracer air that

has properties necessary to trigger cloud is identified as the distributions in the latter converge onto the cloud-base distributions. This allows for the analysis of transport by coherent structures in bulk, but understanding the internal structure (answering *how is this structure carrying out vertical transport?*) and environment (answering *why is cloud-triggering structure forming in this specific location?*) of these coherent structures is very challenging due to number of scalar fields combining to drive vertical transport.

As regards visualizing the spatial extent of these structures this paper presents the first time the spatial extent and physical fields of these structures have been studied in 3D simultaneously. This problem, of understanding the properties of coherent structures defined in one variable (tracer) but understood in other variables (moisture, vertical velocity, &c.) is what drives the current application. In the end result, we anticipate formulation of new physical models describing convection triggering, helping to understand the process and providing better predictions of weather and climate.

5.3 Cartesian Fiber Surfaces



a) Isosurface, fiber surface and Cartesian fiber surface with opacity.

b) Extrusion of the fiber surface control polygon.

Figure 5.3: We intersect the volume contained in the isosurface (sublevel set) of the scalar function h and the volume contained in the fiber surface (inner-level set) of the bivariate combination of two other scalar functions (f, g). To perform the intersection we take another fiber surface with respect to the combined trivariate function (f, g, h). We define the control polyhedron in $\mathbb{R}^3$ as the boundary of the Cartesian product of the interval defining the sublevel set of h and the inside of the polygon defining the inner-level set of (f, g). The resulting shape in three dimensions is an extrusion of the polygon, and its distance fields can be easily computed component wise.

From Section 3.7 we know that fiber surfaces can be computed with the signed distance field of arbitrary geometry in any dimension. However, when the range of a multivariate function is high dimensional, it is difficult to create an interface for setting the fiber surface's control geometry explicitly. We present a method called

Cartesian fiber surfaces, in which we form the control geometry of fiber surfaces from lower dimensional primitives for which we have efficient and robust tools. Furthermore, Cartesian fiber surfaces give us a novel way of rephrasing the problem of geometric intersection for volumes contained in isosurfaces and fiber surfaces.

We will begin our discussion with two base case examples that illustrate the process and then present the general theory. In the first example consider a scalar function $f_1 : \mathbb{R}^3 \to \mathbb{R}$ and an isovalue $h_1 \in \mathbb{R}$. The isovalue separates the range of f_1 in two sets $(-\infty, h_1]$ and $[h_1, \infty)$ whose preimages separate the domain into the well known sublevel and super-level sets. Next, consider a second function f_2 with isovalue h_2 and the intersection of the preimages of the intervals $[h_1, \infty] = I_1$ and $[h_2, \infty) = I_2$ defining the super-level sets.

The intersection of the preimage of I_1 and the preimage of I_2 consists of all points in the domain which satisfy the condition that $f_1(x) \in I_1$ and $f_2(x) \in I_2$. We can combine f_1 and f_2 into a bivariate function with two scalar components $(f_1, f_2)(x) = (f_1(x), f_2(x))$ and evaluate the "and" condition component wise as $(f_1, f_2)(x) \in (I_1, I_2)$. The set of all points (I_1, I_2) is equivalent to the Cartesian product of I_1 and I_2 which is a filled rectangle in the plane. Instead of computing a geometric intersection in $\mathbb{R}^3$ we can compute the fiber surface of the boundary of the rectangle as a control polygon. We call that fiber surface a Cartesian fiber surface.

In the next example consider the bivariate function (f, g) and the scalar function h. From previous work we know that we can extract the fiber surface of arbitrary line segments or subsets of the plane. However, here we're interested in intersecting the volumes contained in separating surfaces. In practice this means fiber surfaces of polygons because those separate the range. Their preimage separates the domain into what we call an inner-level set (all point which are mapped inside the polygon) and an outer-level set (all points mapped outside). The inner and outer level sets are generalizations of the sub and superlevel sets.

As shown in Figure 5.3 we can intersect the super-level set of h and inner-level set of (f, g) with a Cartesian fiber surface. This operation extrudes the polygon in a third dimension in a new $\mathbb{R}^3$ range of a trivariate function (f, g, h). We can apply this more generally to any two multivariate functions and any two separating hypersurfaces in the range. We discuss this in detail in the following section.

We will compute the Cartesian fiber surface using a signed distance field by combining the distance fields of the component isosurfaces and fiber surfaces. Since the distance metric in the Cartesian product space is defined component wise we can compute the signed distance field component wise as well. This means that we can use the Euclidean norm and take the square root of the sum of squared distances given by the component spaces. Once we have the distance field we give it a sign based on whether the point is inside all of the component polytopes or not. This method applies to any metric where combining the minimum distance in every component space yields the minimum distance in the Cartesian product space.

The signed distance field of a Cartesian fiber surface can be computed in linear time, given the signed distance fields of the input fiber surfaces. Then we can use marching

cubes to extract the required geometry. When we're not considering contained volume our method can be used in conjunction with feature level sets Jankowai & Hotz (2018b) to combine multiple traits.

5.3.1 General Cartesian Fiber Surfaces

Working more generally we will expand some of the definitions from Fiber surfaces. First we'd like to define a fiber surface and a fiber surface control polygon when the dimension of the range is higher dimensional than $\mathbb{R}^2$.

Definition 5.1 (Fiber Surface Control Polyhedron (FSCP))**.** *We define a fiber surface control polyhedron in $\mathbb{R}^n$ where $n \geq 2$ to be the triangulation of the continuous image (n-1) dimensional sphere.*

Given a FSCP S we know that $\mathbb{R}^n - S$ has two path-connected components by the Jordan–Brouwer separation theorem. Since S is compact that means that it is a closed and bounded (recall from Section 2.1). Therefore one of the regions in $\mathbb{R}^n - S$ is bounded by S, we will call that region the inner level set of S denoted as S^-. The other region in $\mathbb{R}^n - S$ is not bounded and we will call it the outer level set of S denoted as S^+. Note that since S is closed then $\mathbb{R}^n - S$ is open, but we'll consider both S^- and S^+ to include S so $\partial S^- = \partial S^+ = S$.

Now we will show that the preimage of a fiber surface control polyhedron is closed surfaces that separates the point that map onto the inner and outer level sets.

Lemma 5.1. *Suppose we have a continuous mapping $f : \mathbb{R}^n \to \mathbb{R}^m$ and an FSCP $S \subset \mathbb{R}^m$ in the range. Then $f^{-1}(S)$ is closed surface that separates $\mathbb{R}^3$ in two connected components, namely $f^{-1}(S^-) \cup f^{-1}(S^+) = \mathbb{R}^3$ and $f^{-1}(S^-) \cap f^{-1}(S^+) = S$.*

Proof. We will reduce this to the scalar case and use what we know about isosurfaces to complete the proof. Let us define the signed distance field of S as $g : \mathbb{R}^n \to \mathbb{R}$ such that $d(x) = min_{y \in S}(d(x, y))$ if $x \in S^+$ and $d(x) = -min_{y \in S}(d(x, y))$ if $x \in S^-$. Then the composition $(g \circ f) : \mathbb{R}^3 \to \mathbb{R}$ is a scalar field. Given how we've defined g we have that $g(S) = 0$, $g(S^-) \leq 0$ and $g(S^+) \geq 0$. Therefore $(g \circ f)^{-1}(0)$ is an isosurface and so it is a separating surface between the sets $(g \circ f)^{-1}((-\infty, 0))$ and $(g \circ f)^{-1}((0, \infty))$ Wenger (2013b). However $(g \circ f)^{-1}((0, \infty)) = f^{-1}(S^+ - S)$ and $(g \circ f)^{-1}((-\infty, 0)) = f^{-1}(S^- - S)$ which completes the proof. □

As a corollary we have that $f^{-1}(\partial S^-) = f^{-1}(S) = \partial f^{-1}(S^-)$ for any FSCP S. This analogously applies to S^+ as well. Using this corollary we can finally show how Cartesian fiber surfaces are defined in general using the following lemma.

Lemma 5.2. *Suppose we have a continuous mapping $f : \mathbb{R}^3 \to \mathbb{R}^m$ and a continuous mapping a continuous mapping $g : \mathbb{R}^3 \to \mathbb{R}^n$. Further suppose we have two fiber surface*

control polyhedrons $A \subset \mathbb{R}^m$ and $B \subset \mathbb{R}^n$. Then we have that $\partial(f^{-1}(A) \cap g^{-1}(B)) = (f,g)^{-1}(\partial(A \times B))$.

Proof. First we will show that $f^{-1}(A) \cap g^{-1}(B) = (f,g)^{-1}(A \times B)$ with the following series of equalities.

$$f^{-1}(A) \cap g^{-1}(B) =$$
$$= \{x \in \mathbb{R}^3 \mid f(x) \in A\} \cap \{x \in \mathbb{R}^3 \mid g(x) \in B\} =$$
$$= \{x \in \mathbb{R}^3 \mid f(x) \in A \text{ and } g(x) \in B\} =$$
$$= \{x \in \mathbb{R}^3 \mid (f,g)(x) \in A \times B, \text{ where } A \times B \subset \mathbb{R}^{m+n}\} =$$
$$= (f,g)^{-1}(A \times B)$$

Once we take the boundary of both sides we have that: $\partial(f^{-1}(A) \cap g^{-1}(B)) = \partial((f,g)^{-1}(A \times B))$ Since the Cartesian product of the two closed balls is a closed ball in $\mathbb{R}^{n+m}$, the corollary of Lemma 5.1 implies that $(f,g)^{-1}(\partial(A \times B)) = \partial(f,g)^{-1}((A \times B))$. Finally we have obtained that $\partial(f^{-1}(A) \cap g^{-1}(B)) = (f,g)^{-1}(\partial(A \times B))$ which completes the proof. □

In general we can take the intersection of the inner level sets of more than two fiber surfaces by combining them into a single Cartesian fiber surface. This introduces a way of querying data where we can easily chain together multiple "and" conditions and evaluate them. Each condition defines a feature via the inner level set of an isosurface or a fibers surface. Next we are going to show that we can compute Cartesian fiber surfaces efficiently.

5.3.2 Computation

To compute a fiber surface we consider the signed distance field of the FSCP and compute an isosurface at zero. Here we will show how to compute a Cartesian fiber surface by combining the signed distance fields of two other fiber surfaces. The signed distance field of a Cartesian fiber surface can be computed in linear time, given that we have the signed distance fields of the fiber surfaces.

Consider the two multivariate functions from we have been using so far $f : \mathbb{R}^3 \to \mathbb{R}^n$ and $g : \mathbb{R}^3 \to \mathbb{R}^m$ and their combination $(f,g) : \mathbb{R}^3 \to \mathbb{R}^{n+m}$. We can compute the distance between two points $x = (x_1, .., x_{n+m})$ and $y = (y_1, .., y_{n+m})$ in $\mathbb{R}^{n+m}$ using the two distances of their projections in $\mathbb{R}^n$ and $\mathbb{R}^m$. We have that

$$d(x,y) = \sqrt{\sum_{i=0}^{n+m} (x_i - y_i)^2} = \sqrt{\sum_{i=0}^{n} (x_i - y_i)^2 + \sum_{i=n+1}^{m+n} (x_i - y_i)^2} \tag{5.1}$$

where the two sums in the square root are the squared distance in projection.

In order to compute a signed distance field using this we introduce the following lemma.

Lemma 5.3. *Given the two signed distance fields d_A and d_B of two FSCP $A \subset \mathbb{R}^n$ and $B \subset \mathbb{R}^m$ their combined signed distance field $d_{A\times B}$ can be expressed as:*

$$d_{A\times B}(x_1, \dots x_{n+m}) = \sqrt{d_A(x_1, \dots, x_n)^2 + d_B(x_{n+1}, \dots, x_{n+m})^2}.$$

Proof. Given $(x_1, \dots, x_{n+m})$ we define $\pi_A : \mathbb{R}^{n+m} \to \mathbb{R}^n$ as the projection $\pi_A((x_1, \dots, x_{n+m})) = (x_1, \dots, x_n)$ and $\pi_B : \mathbb{R}^{n+m} \to \mathbb{R}^m$ as the projection $\pi_B((x_1, \dots, x_{n+m})) = (x_{n+1}, \dots, x_{n+m})$.

We present the following equalities and then explain why they hold. The Cartesian product of two compact sets is also compact. All the inf values in the equations below exists because we assume all our sets are compact.

$$d_{A\times B}(x) = \inf_{y\in A\times B} \{d_{A\times B}(x, y))\} =$$

$$= \inf_{\pi_A(y)\in A \text{ and } \pi_B(y)\in B} \left\{ \sqrt{d_A(\pi_A(x), \pi_A(y))^2 + d_B(\pi_B(x), \pi_B(y))^2} \right\} = \tag{5.2}$$

$$= \sqrt{\inf_{\pi_A(y)\in A \text{ and } \pi_B(y)\in B} \left\{ d_A(\pi_A(x), \pi_A(y))^2 + d_B(\pi_B(x), \pi_B(y))^2 \right\}} = \tag{5.3}$$

$$= \sqrt{\inf_{\pi_A(y)\in A} \left\{ d_A(\pi_A(x), \pi_A(y))^2 \right\} + \inf_{\pi_B(y)\in B} \left\{ d_B(\pi_B(x), \pi_B(y))^2 \right\}} \tag{5.4}$$

$$= \sqrt{\left\{ \inf_{\pi_A(y)\in A} d_A(\pi_A(x), \pi_A(y)) \right\}^2 + \left\{ \inf_{\pi_B(y)\in B} d_B(\pi_B(x), \pi_B(y)) \right\}^2} \tag{5.5}$$

$$= \sqrt{d_A(\pi_A(x))^2 + d_B(\pi_B(x))^2}.$$

Equation 5.2 comes from Equation 5.1. Equation 5.3 commutes the square root with the inf because the square root is a monotonely increasing function. Equation 5.4 uses the basic property of inf that $\inf(A + B) = \inf(A) + \inf(B)$. The last equation (Equation 5.5) commutes the square with the inf because the square is a monotonely increasing function on the positive numbers where distance is defined. The result then follows from the definition of a distance field in the two factor spaces. □

5.4 Tracer Visualiser Application

In this section we will discuss the application we have implemented to aid scientists in visualising convective cloud formation by understand the physical properties of cloud triggering objects defined by tracer (recall Section 5.2). We call the application the Tracer Visualiser. First we pose the following application requirements.

5.4.1 Application Requirements

The key issue for domain scientists is to understand how to define cloud triggering coherent structures in the planetary boundary layer directly from the physical fields available (such as moisture, velocity, temperature). In order to continue their research experts require a tool that supports the following requirements:

R1) Spatial visualization of features, defined by the tracer.

R2) Visualization of moisture, heat and velocity in features.

R3) Interactive exploration of features and their attributes.

R4) Attribute based spacial queries and feature refinement.

R5) Suppression of small scale features such as noise.

R6) Geometric export of features for further quantification.

5.4.2 Application Design

From our literature review in Part 3 we know that tools for scalar fields are now well understood, while tools for bivariate fields are in development. We propose to combine those tools in a trivariate visualisation application to study relationships between features defined in one variable and features defined in two others. The features defined in one variable will be based on a decaying passive tracer scalar field resulting from a numerical simulation. As we have shown in Subsection 5.2.1 the tracer field is accepted by the domain experts as a good approximation of cloud triggering air structures. We will study the attributes of those features in relation to the underlying physical scalar fields of temperature, moisture and vertical velocity.

The first requirement (R1, refer to Subsection 5.4.1) of our application is to render the features (or air parcels) defined by the tracer scalar field. Those features are the connected components of the super-level set of the domain at a user-defined threshold. We render the boundary of the super-level set using an isosurface and identify the connected components using the join tree (recall Section 3.4). In order to suppress small scale features and noise (R5) we use topological simplification based on persistence (recall Section 3.5) or contained volume.

The second requirement is to visualise the moisture, heat and vertical velocity of those features (R2). We visualise the joint distribution of those scalar attributes in

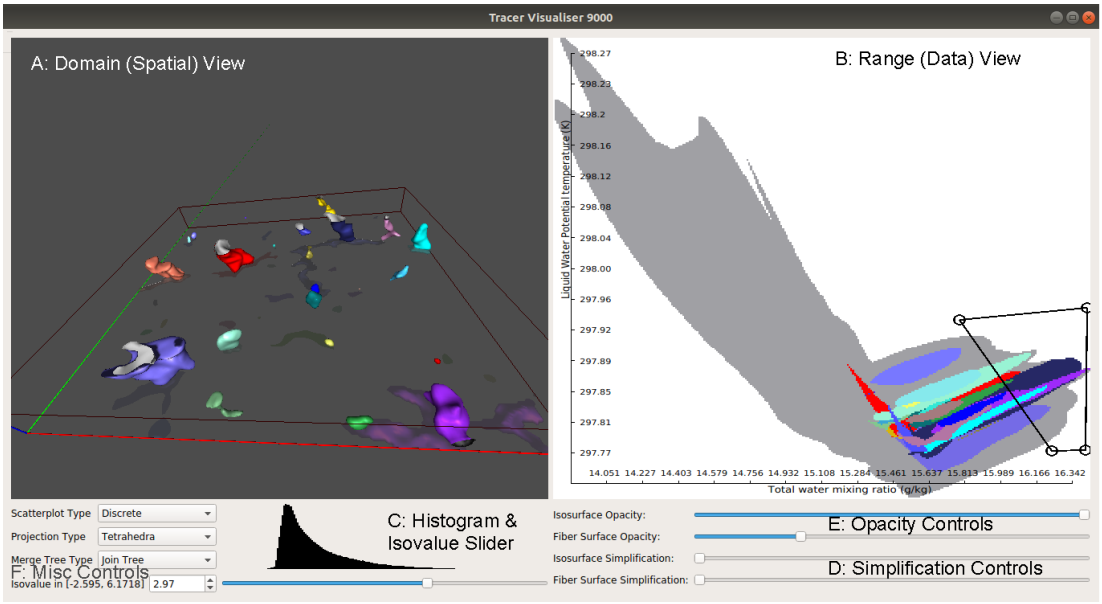


Figure 5.4: The user interface of the tracer visualiser application. It is based on a main (domain) view (A), a data (range) view (B), and ancillary controls (C)-(E).

pairs with three continuous scatterplots (recall Section 3.7). We use the scatterplots of the whole domain as a canvas on which we project the features to study their shape in attribute space. The features are projected by computing their individual continuous scatterplots (as subvolumes of the domain) and compositing them together, much like a collage.

To enable interactive exploration of the features and their attributes (R3) we allow for brushing and linking via fiber surfaces. The user can brush the scatterplot with a fiber surface control polygon to highlight an area of interest and then link back to the domain by rendering the fiber surface of the polygon. The fiber surface reveals other parts of the spatial domain, outside the features, which have similar properties. This is useful in investigating hypotheses about the environment in which convective clouds are triggered.

It is important to the user to study the internal structure of their features - how they're composed of the physical attributes. To enable this we support feature refinement (R4) by rendering the parts of features that have particular properties inside the highlighted area in the scatterplot. Determining this geometrically is equivalent to taking the intersection of the volume contained in the isosurface and the volume contained in the fiber surface. In order to avoid the costly computation of a geometric intersection we use the novel technique we introduced called Cartesian fiber surfaces (recall Section 5.3).

5.4.3 Application Implementation

The design of the user interface flows from the following: a shared spatial representation of features in the domain, supported by secondary views displaying data attributes. This means a secondary view for a continuous scatterplot or an equivalent representation of bivariate data, and a separate view for a histogram for the single scalar field. Our input data consists of three scalar fields defined over a time-varying three dimensional rectilinear grid. See an example of our user interface in Figure 5.4.

Domain View

In our main spatial view, we show the domain of the function, the isosurface, representing the boundary of the super-level set and the fiber surface, representing the boundary of areas that correspond to specific range attributes. Displaying both the isosurface and the fiber surface at the same time often results in excessive visual clutter, making it difficult to determine the relationship between two. Even with opacity controls it is not aways possible to identify the intersection between the volumes contained in both surfaces. This is why we also render a third surface, the boundary of the intersection of those volumes, using Cartesian fiber surfaces (described in detail in Section 5.3).

To support interactive exploration, we use the color channel to distinguish between the connected components of the isosurface, the fiber surface and the Cartesian fiber surface. Since our domain experts are not color-blind, we have not at present implemented colormap selection, but this is clearly important if we wish to distribute the tool for others to use, and we expect to support it at a later date. Our application also supports feature selection via a left mouse click. The selected object is highlighted in the domain view and its projection is highlighted in the range view. Upon object selection we focus the camera on the centroid of its triangles across all timesteps to ensure smooth transition between frames.

We also know that many features are too small to be of interest, so we support simplification based on topological persistence. The user selects an isovalue of interest and a simplification threshold. The main view then shows all isosurfaces whose associated super-level sets have persistence greater than the threshold, using the standard merge tree computation Carr *et al.* (2003). Fiber surfaces and Cartesian fiber surfaces are simplified by contained volume.

Attribute View

In the attribute view, we show the probability distribution function of the two secondary variables with the continuous scatterplot, which we compute using the Topology Toolkit ttk (2020). The continuous scatterplot is like a canvas on which we render the projections of the connected components of the super-level set. In order to compute the projections we remesh the domain along the isosurface boundary; identify connected components using the join tree; compute their individual continuous scatterplots and finally composite them onto the continuous scatterplot of the whole domain. For con-

sistency between the domain and the attribute view we use the same colour for the features and their projection.

At this stage, an additional problem arises - choosing an appropriate colour map for the scatterplot and the projected features. If we use a coloured heatmap (as the original continuous scatterplot approach Bachthaler & Weiskopf (2008b)) it would be difficult to distinguish between the features and especially their boundary. Additionally if a feature is small, its scatterplot will have low density relative to the overall plot. In that case only the feature's highest density regions will have enough color contrast for visibility. We therefore chose a simple binary colormap (grey-white) for the continuous scatterplot and colored binary colormaps for projecting individual features.

In practice there is a high degree of overlap between the projected features, so we allow the user to render them with some transparency and bring them to the top when they are selected in the domain view. The shape of the projections allows the user to form hypotheses about the attributes of features. We allow brushing via a fiber surface control polygon for the user to verify those hypotheses and link to the domain view with a fiber surface. The user can modify the fiber surface control polygon by adding, removing and translating points.

Performance and Design Considerations

The most computationally expensive part of our application was rendering the continuous scatterplot and the feature projections. This prohibited interactive exploration of high resolution data sets with many timesteps. For some scatterplots the binary colormap was not sufficient and we needed lower opacity for low density areas. However that required that the user be familiar with the continuous scatterplot algorithm and the opacity function design.

Therefore we also supported the use of the discrete scatterplot for the whole data and feature projections. The two main reasons were efficiency and reducing the user's overhead with a familir visualisation that is more intuitive for them. In practice, we typically worked with the discrete scatterplot, but periodically cross-checked the results against the correct continuous scatterplot to ensure that our conclusions were accurate.

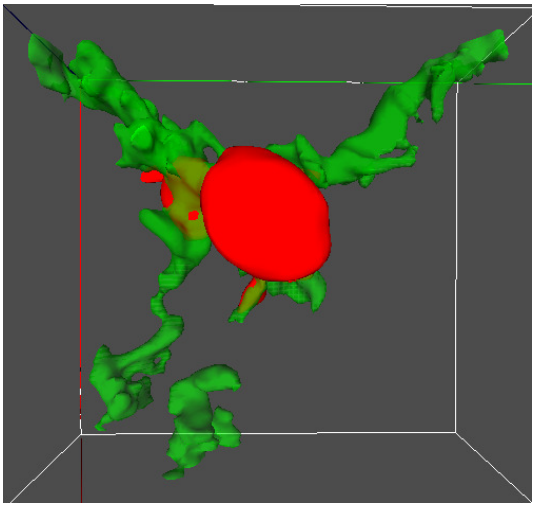
The speed and smoothness of transitions between timesteps is of key importance for interactive visualisation. Analogous to key-frame animation the user would like to be able to quickly flip through the timeslices and observe the change in features' geometry, their projection, the fiber surface and Cartesian fiber surface. When the user changes a parameter, like the isovalue or fiber surface control polygon, we recompute the resulting meshes, projections and intersections for all timesteps in parallel using OpenMP.

5.5 Case Study: Convective Triggering

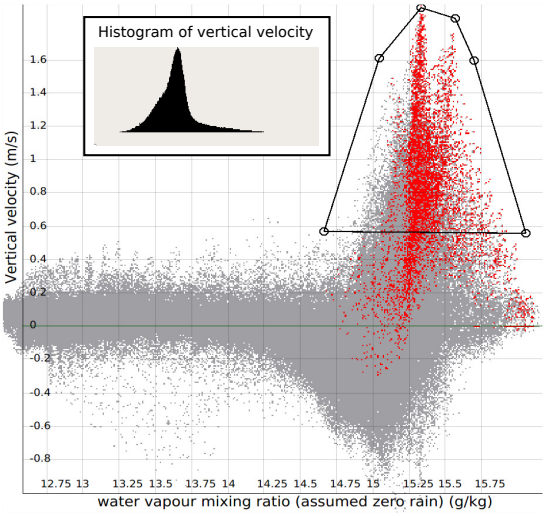
In this section we describe the workflow and insight gained from using the application. We are interested in guiding the development of a physical model describing the properties and evolution of individual cloud-triggering features. We give two examples.

The first example is focussed on understanding the structure of the environment in which a feature forms and rises, enabling the formulation of hypotheses around why features form in specific regions. The second is focussed on understanding the internal structure of a cloud-triggering feature, which enables formulation of a physical model of how vertical transport is carried out by these structures.

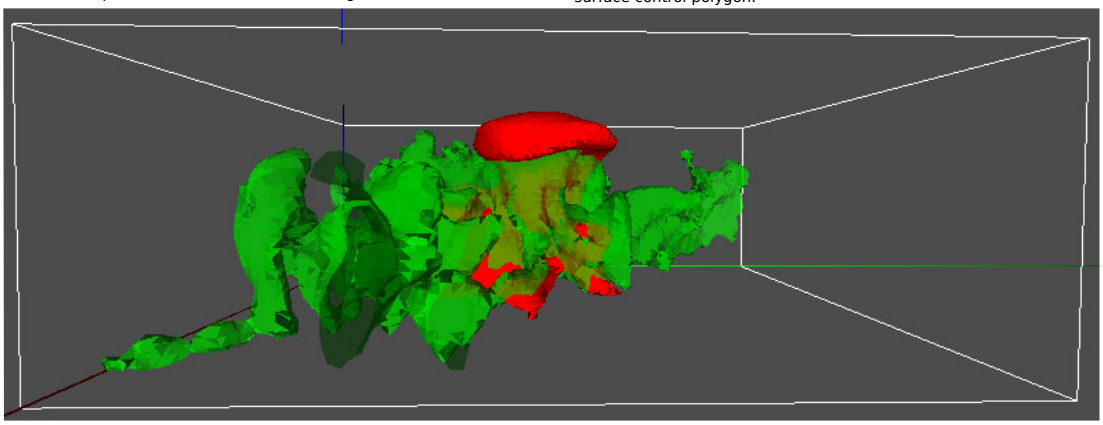
We examine three fields relevant to triggering convective clouds: the vertical velocity w, the specific concentration of water vapour q measuring the mass of water vapour per mass of mixture and the potential temperature θ, which is the absolute temperature corrected for the drop in temperature under adiabatic expansion, allowing for easy comparison of temperatures at different heights in the boundary layer. For convenience, we will refer to θ as "temperature". Both the q and θ fields display the physical characteristic that in the absence of mixing (and condensation, which does not occur below cloud base) they will both be *conserved*, meaning that the value will not change as a volume of air rises.



a) Top down view of the isosurface (red) and fiber surface (green). Shows three parts of the fiber surface meeting at the isosurface.



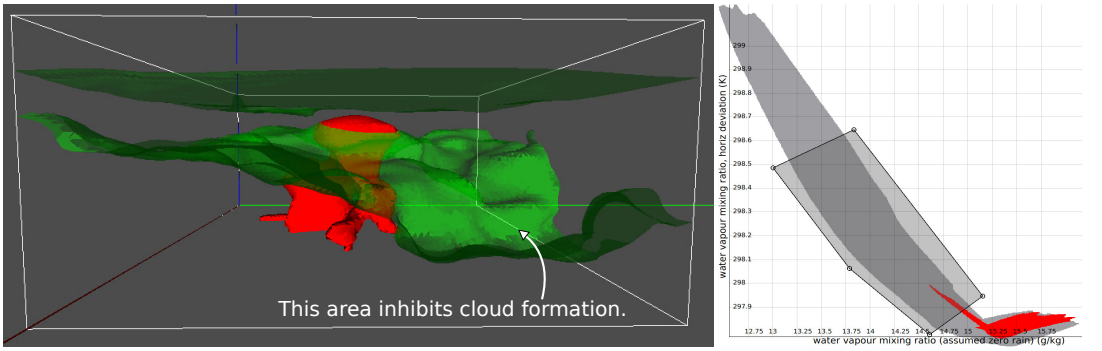
b) Discrete scatter plot (gray) with feature projection (red) and a fiber surface control polygon.



c) Side view of the isosurface (red) and fiber surface (green). Shows the vertical contribution of the fiber surface to the isosurface.

Figure 5.5: Characterising the environment in which cloud-triggering features (red surface in *a)* and *c*) in scatterplot *b)*) form based on the projection of the feature in the scatterplot of vertical velocity and water vapour concentration (the product of which is vertical moisture transport). The projection of the feature (in *b)*) shows that it mainly contains rising moist air. A fiber surface (green surface in *c)*) of the area of the scatterplot with similar properties, the moistest air parcels which are also rising, shows that the structure is embedded within and at the nodal-point of three horizontal segments of rising moist air.

The test case shows the development of shallow convection over ocean based on the RICO campaign vanZanten *et al.* (2011) modified to remove ambient windshear and to fix surface moisture and heat fluxes to $F_l = 150\text{W/m}^2$ and $F_s = 7\text{W/m}^2$. The UCLALES Large-Eddy model Stevens *et al.* (1999) was used to model the case on a double-periodic domain of 20km $\times$ 20km $\times$ 4km at an isotropic grid resolution of



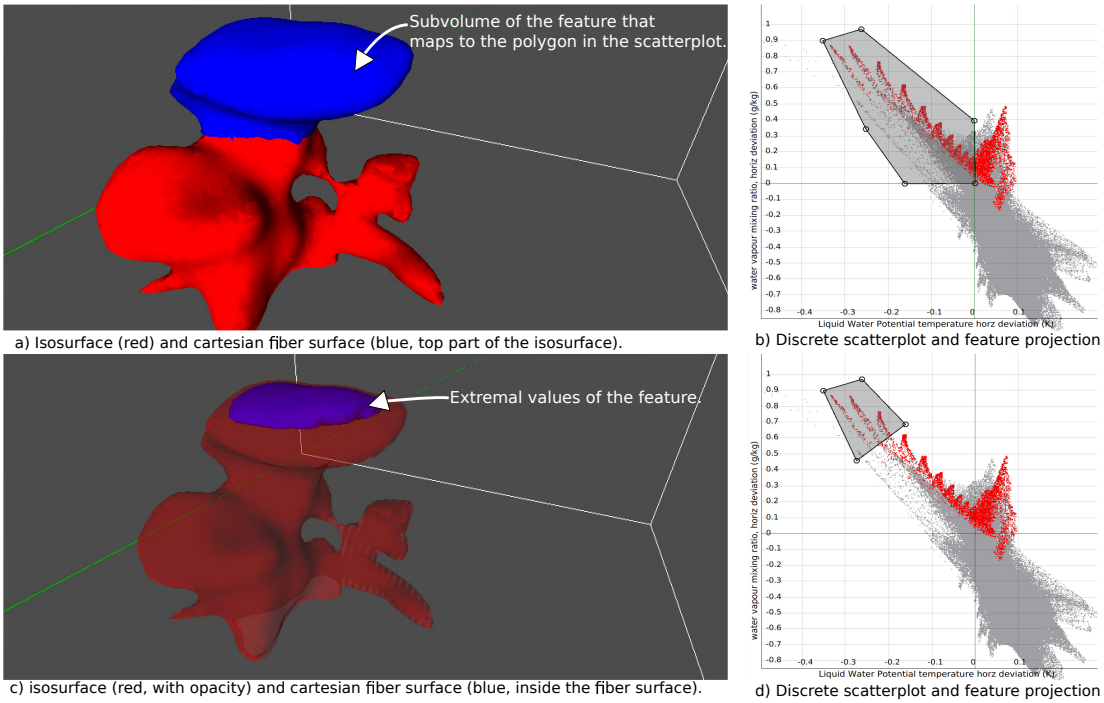
a) Isosurface of the cloud-triggering feature (red) and fiber surface (two components in green).

b) Continous scatter plot (gray) and feature projection (red).

Figure 5.6: Characterising the environment in which cloud-triggering features form based on the projection of the feature in the scatterplot of temperature and water vapour (both of which contribute to density). The projection of the feature (red surface in *a)*) in scatterplot *b)*) shows that it is comprised of moist-cold and moist-warm linear segments. Rendering a fiber surface (green surface in *a)*) of the warmest and driest part of the scatterplot distribution demonstrates how the surface of constant density in the environment is curved towards the bottom part of the domain, inhibiting formation of cloud-triggering structures there.

$\Delta x = 25$m, with 3D fields of water vapour concentration, potential temperature and vertical velocity output every $\Delta t = 1$min. Instead of considering the value of moisture (q) and temperature (θ) at every point in the boundary layer we may instead consider the deviation from the horizontal mean, e.g. $\theta'(x, y, z) = \theta(x, y, z) - \overline{\theta}(z)$. This corrects for the fact that the moisture and heat sources are at the ocean surface and thus both fields have a larger horizontal mean value near the ocean. From the simulation output 2km $\times$ 2km regions where individual boundary layer structures rise and trigger isolated convective clouds were identified using visualisations of the tracer field height and clouds (Figure 5.2 top right). These smaller 3D regions were then extracted into individual data-files which were fed into the visualiser application.

In the first example we focus on characterising the environment in which cloud-triggering features form. We examine the vertical velocity and moisture (water vapour) concentration variables. The product of water vapour specific concentration and vertical velocity (together with mixture density which varies much less) gives the moisture flux and so looking at the covariance of these variables enables us to examine the vertical moisture transport, which is necessary for triggering convective clouds. Projecting the features identified by the existing (isosurface) method into the scatterplot of vertical velocity and moisture reveals that the cloud-triggering structures contain the moistest air masses and mainly air that is rising, but also some subsiding (negative vertical velocity) air. This is overturning air in vortex rings that is embedded in the features (Figure 5.5). The features exist primarily inside an isolated peak of positive



a) Isosurface (red) and cartesian fiber surface (blue, top part of the isosurface).

b) Discrete scatterplot and feature projection

c) isosurface (red, with opacity) and cartesian fiber surface (blue, inside the fiber surface).

d) Discrete scatterplot and feature projection

Figure 5.7: Decomposing a cloud triggering feature (super-level set component) using the volume contained in a fiber surface using Cartesian fiber surfaces *a)* and *c)* of two concentric fiber surface polygons in the scatterplot *b)*. The Cartesian fiber surface intersections (blue in *a)* and *c)*) reveals that the top of the feature is cool and moist (fiber surface control polygon in *b)*), with cooler and moister regions (fiber surface polygon in *d)*) located concentrically towards the topmost edge of the feature.

vertical velocity and large moisture content in the scatter-plot, a peak which is not visible when looking only at the histogram of vertical velocity (see inset of Figure 5.5) due to the large number of points where air is hardly moving vertically. Drawing a fiber surface control polygon around this peak we see that the cloud-triggering features are completely embedded within regions which contain rising moist air, even though the features locally contain subsiding air. We also see that the cloud-triggering feature has formed at the nodal-point of three horizontal segments of rising moist air, this was observed across several examples suggesting these nodal intersections of rising moist air are favourable for creating cloud-triggering structures. Using fiber surface control polygons to segment the projected feature we found it to be comprised of two discrete components, below-cloud tracking the moistest rising datapoints and in-cloud decreasing with vertical velocity and water vapour concentration. This suggests that a combination of vertical velocity and water vapour concentration may form a way to identify cloud-triggering structures.

Further examining the environment of a cloud-triggering feature we look at the

potential temperature and water vapour concentration fields as these together define the density, and regions with lighter environment air form a barrier for a rising feature as the lighter environment necessitates that the feature be even lighter to continue rising (Figure 5.6 *b)*). Above the height at which clouds form there is a monotonic relationship between temperature and water vapour with height, so that potential temperature increasing with height and water vapour decreasing. Drawing a fiber-surface control polygon across this region of monotonic relationship in the scatter plot we see that as cloud-base height is approached the fiber-surfaces transition from being at near-constant height across the domain to being pulled downwards toward the ground in the region where no cloud-triggering structures have formed. This demonstrates how surfaces of constant density in the environment inhibit cloud-formation in regions of the horizontal domain.

In the second example we examine the internal structure of a cloud-triggering feature (Figure 5.7) by considering the local anomaly of potential temperature and water vapour relative to the horizontal mean. By subtracting the horizontal mean we can examine how the feature is doing transport against the vertical mean gradient in both temperature and moisture. The cloud-triggering feature (as identified by the existing isosurface technique) is seen to contain both air masses that are cold and moist relative to the environment and warm and dry relative the environment. By drawing fiber-surface control polygons in the scatter plot we can build up a structural representation of the cloud-triggering feature. The fiber-surface and the isosurface feature indicates (not shown) that the top of the cloud-triggering feature is embedded in an elevated region with high moisture content, which may help the feature avoid dissipating by mixing with dry air. By using the Cartesian fiber surface intersection with the isosurface we can see that the top of the feature consists of concentric layers increasingly cold and moist regions, with the top-most disc-like region of the feature being the moistest and coolest. Elucidating this internal structure is key to forming a mathematical model of these cloud-triggering features.

These insights enable us to develop hypotheses for the mechanisms and the necessary environmental conditions for clouds to be formed. Specifically, they help us guide how to define a cloud-triggering feature and what aspect of these features to track when developing further algorithms to carry out quantitative analysis. In addition the example feature examined in this section is that of an isolated cloud-triggering feature leading to the formation one cloud with a single buoyant core, however clouds often appear to form from multiple cloud-triggering features in quick succession with a number of convective growths within the same cloudy area. Having the tool described in this work will enable the study of the spatial interactions in this more complicated case in further work, allowing us to build hypothesis for the necessary conditions for these more complicated multi-core clouds to form.

Part IV

Reeb Spaces

Chapter 6

Reeb Space Local Neighborhood Classification

6.1 Introduction

In previous chapters we introduced an important tool for scalar field visualisation called an isosurface and its associated topological data structure called the Reeb graph. However, in analysing real world data there is usually more than one property of interest. Hence, we introduced a bivariate visualisation tool called the fiber surface. The topological data structure associated with the fiber surface is a generalisation of the Reeb graph called the Reeb space. The Reeb space is far less studied and understood than the Reeb graph and it is the object of investigation of this part.

Given a scalar function over a three dimensional volume, the Reeb graph is a one dimensional topological object. The Reeb graph is made up of vertices and edges that connect them. Given a bivariate function over a three dimensional volume, the Reeb space is a two dimensional object. The Reeb space is a polyhedron made up of vertices, edges connecting the vertices and two dimensional sheets connecting the edges.

The Reeb space is far more complex to understand and visualise than the Reeb graph because of the intricate ways in which sheets can attach to one another. Fully understanding the structure of the Reeb space is key to developing efficient algorithms for computing, simplifying and using it. Previous work has laid the foundation of understanding the local structure of the Reeb space, but only for a restricted case of PL maps. This limits the applicability to practical data analysis.

The key contribution of this part is a fully combinatorial method for computing all possible local structures in a Reeb space, parametrized by the degree of the vertices and edges of the input mesh. Our method makes no assumptions about the input mesh aside from simple conditions on the data values which can be guaranteed by standard perturbation techniques such as simulation of simplicity. We demonstrate our method by giving a full classification of all Reeb spaces neighbourhoods of input meshes with

low vertex and edge degree.

We begin with a literature review of the previous work done related Reeb spaces in Section 6.2. Then we present the methodology we will use for describing Reeb space neighbourhoods in Section 6.3. After that we give our assumptions in Section 6.4. The main theoretical contributions are given in Sections 7.1, 7.2 and 7.3 where we describe the local topology of PL fibers. In Section 8.1 we discuss the global topology of PL fibers and present the intuition behind computing Reeb space neighbourhoods. Following that in Section 8.2 we present our algorithms for enumerating and generating all possible Reeb space neighbourhoods for meshes with bounded vertex and edge degree. In Section 9.1 we describe our implementations of those algorithms. Finally we present the results of our Reeb space neighbourhood classification in Section 9.2.

6.2 Background

One of the first papers that investigates the singularities of mappings beyond the scalar case is by Whitney (1955). Whitney studied smooth mappings of the plane into the plane called generic maps. The paper showed that generic maps (which is another form of stability Section 2.3) have simple behaviour depending on the rank of the Jacobian. That behaviour is described by the singularities (or set of singular points) of the stable maps, which consists of one dimensional curves called folds and isolated points called cusps, which can connect some folds. Furthermore these generic maps are dense (see Definition 2.1) in the set of all smooth maps from the plane to the plane.

The results of Whitney were later extended by Levine Levine (1966) by examining mappings of manifolds to the plane. Furthermore Èliašberg (1970) generalised these results further by looking at mappings between manifolds, but only their folding singularities (see Section 2.3). The following book is a comprehensive modern introduction on Fiber Topology Saeki (2004). An introduction for computer scientists can be found in Saeki (2017).

Stable maps and their singularities are often studied using a quotient space known as the Stein factorization, which we know as the Reeb space. One of the first descriptions of the Reeb space and its local neighbourhoods is given in Levine (1988). The author describes the local connections of sheets by studying singular fibers. However, the author only presents results for orientable manifolds. An important results about the Reeb space is that it can be triangulated for a large class of functions Hiratuka & Saeki (2013). Triangulation is important because it ensures that computing the Reeb space is computationally feasible. There have also been results concerning the local Reeb space neighbourhoods and simplification for manifolds with boundary Saeki & Yamamoto (2016a,b).

In the PL setting the singular set of a mappings has been defined as the Jacobi set Edelsbrunner & Harer (2002b). The Reeb space for PL maps has been defined in Edelsbrunner *et al.* (2008c) where the authors give a method for the description of Reeb space neighbourhoods and a generic Reeb space algorithm. However the authors do not provide a practical implementation of the algorithm. The first implementation

for the $\mathbb{R}^3 \to \mathbb{R}^2$ case is given in Tierny & Carr (2017b). In the next sections we will expand the details of these works on PL Reeb spaces.

Smooth Reeb Space Neighborhood

In fiber topology the study of singular fibers is closely related to the study of local Reeb space neighbourhoods. The way that local neighbourhoods of the Reeb space are described is by first classifying the singular fibers of a mapping Saeki (2004). The first step in classifying singular fibers is to describe the local neighbourhood of a singular fiber around the singular point. Then we extend the curves in that local neighbourhood to the whole domain to connect them globally. The way that the curves connect, with restrictions enforced from the stability of the mapping, tells us how the singular fiber changes the connected components of the regular fibers in its neighbourhood. Since the Reeb space encodes the connectivity of those fibers, this also completely describes the Reeb space neighbourhood as well.

Jacobi Sets

The Jacobi set is the PL analogue of the singular set of points of a smooth function Edelsbrunner & Harer (2002b). For the approximation of two Morse function by a bivariate PL map the Jacobi set is a set of connected edges. To determine whether an edge of the combinatorial manifold is Jacobi we apply a local test. Since this is an integral part of our further investigation we will describe the test in the particular case of 3-manifolds (a similar description can also be found in Tierny & Carr (2017b)).

Given an edge ab, the link of the edge is a triangulation of a circle on n vertices. For a bivariate PL map (f, g) we consider the line defined by the image of the edge $(f, g)(ab)$ in $\mathbb{R}^2$. We then count the connected components of the upper and lower link of ab. The upper link is defined by all vertices and edges and are entirely on one side of the line defined by $(f, g)(ab)$. The lower link is defined likewise for all vertices and edges of the link on the other side of the line defined by $(f, g)(ab)$.

If both the lower and upper link have one connected components the edge is regular, the fiber does not change as it crosses the edge. If the lower link has zero components and the upper link has one component, then the fiber around the edge is a circle, which shrinks to a point. If both the lower link and upper link have more than one component, then locally multiple fiber components merge and immediately split when crossing the edge.

PL Reeb Space Neighborhood

The key result from Edelsbrunner *et al.* (2008c) that enables the description of Reeb space neighbourhoods is the Cone Neighborhood theorem. According to the Cone Neighborhood theorem every point in the Reeb space has a neighbourhood that is a cone over a Reeb space of one dimension less. Given a point in the Reeb space p we take

a small closed disk D around p. We describe the neighbourhood of p by investigating the boundary of the disk ∂D, which is a circle. We call that a walk around p.

Let $(\partial D)^{-1}$ be the connected component that contains p of the preimage of ∂D which is $(f,g)^{-1}(\partial D)$. Then we can take the restriction $(f,g)_{(\partial D)^{-1}}$ of the bivariate function (f,g), which is a scalar function to the unit circle. The cone of the Reeb graph of the restricted function and the center point p gives us the local Reeb space structure around p. Therefore the local Reeb space structure of p is entirely described by the Reeb graph of the circle.

In practise the authors of Edelsbrunner *et al.* (2008c) describe Reeb space neighbourhoods of PL functions by describing the number of connected components over the course of a walk around the point p. We show how we extend that methodology in the following section.

6.3 Method Overview

The methodology we going to us for describing Reeb space neighbourhoods follows previous methods from the smooth Levine (1988) and PL Edelsbrunner *et al.* (2008c) cases. We present our two step process in Figure 6.1

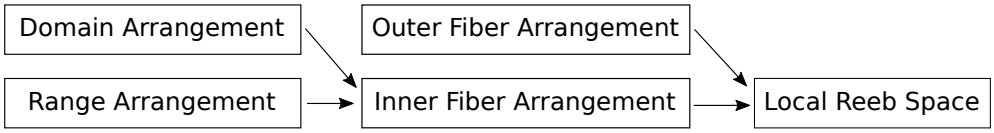


Figure 6.1: A diagram of how a local Reeb space neighbourhood is computed.

In the first step we describe the local behaviour of fibers within a neighbourhood of the point of interest. The way we describe their behaviour is by tracking their connected components and how they appear, disappear, merge and split over the course of a small enough walk around that point. In order to describe the fibers locally we use the domain arrangement and the range arrangement.

The domain arrangement is a description of the neighbourhood of the point in the domain. In the PL setting by neighbourhood we mean the star of the simplex that the point belongs to. In our case that would be a vertex or an edge. Recall (Section 2.4) that we not differentiate between the terms star and closed star, we assume that the star contains all its faces. By range arrangement we mean a combinatorial description of how the image of the vertices of the link of the simplex are arranged around the image of the simplex in the plane.

The second step is to describe the global behaviour of the fiber by extending the local fiber behaviour via the outer fiber arrangement. The outer fiber arrangement describes all the possible ways in which the fiber can be connected outside the star, in the rest of the mesh. Once we have the global fiber behaviour we can compute the Reeb space neighbourhood by tracking the fiber components over the walk.

6.4 Assumptions and Stability

We will assume the domain is a closed (without a boundary and compact) combinatorial 3-manifold and call it M. We will also refer to M as the mesh. Assume we have a bivariate PL map from the mesh onto the plane $(f, g) : M \to \mathbb{R}^2$ where f and g are scalar PL maps.

The only paper in the literature so far that describes Reeb space neighbourhoods of PL maps is Edelsbrunner *et al.* (2008c). In order to describe the neighbourhoods however the authors impose certain restrictions on the PL map that make them less suitable for practical data analysis.

The first restriction is that all Jacobi edges are simple - that means only two fiber components can meet there. More than two fiber components meeting at a Jacobi edge is analogous to monkey saddles in the scalar case Edelsbrunner & Harer (2010). This requirement is resolved as in Edelsbrunner & Harer (2002b) where the authors note that we can break up non-simple Jacobi edges into multiple simple Jacobi edges. However this requires the input mesh to be modified.

The second requirement is that the Jacobi set is a set of PL curves. This is a restriction on the topological structure derived from the data, not directly on the data itself. In practise when we first obtain a dataset, we cannot know a priori what the resulting topological structure will be without computing it. Instead it would be more practical to work with restrictions which are based entirely on the data values. Hence we use the following definition of generic.

Definition 6.1. *A generic bivariate PL map (f, g) is a bivariate PL map such that:*

- *The mapping is injective on the vertices, edges and triangles of the input mesh (analogous to generic Edelsbrunner* et al. *(2008c)) .*
- *No more than two edges meet at a point (unless it's a vertex).*
- *No more than two vertices meet at an edge.*

All three requirements follow the general idea that no small perturbations in the data should change the structure of the Jacobi set or the Reeb space. Indeed, if three edges meet at a point any small perturbation on any of the vertices or any of the edges changes that configuration. The same holds for three vertices meeting on an edge.

Finally, an important question is how we guarantee this condition on any type of input data. In the scalar case we apply symbolic perturbation of the data with the technique simulation of simplicity Edelsbrunner & Mücke (1990). We can do the same in the bivariate case by perturbing the two scalar fields individually Tierny & Carr (2017a).

Chapter 7

Local Fibers

7.1 Local Fibers in a Tetrahedron

We will start our investigation by demonstrating the properties of a fiber in a single tetrahedron and consequently in two tetrahedra sharing a triangle. These cases are regular, which means that no topological change happens to the fibers. they are topologically equivalent to a line segment. However we use these regular cases as building blocks to understand the more complex singular cases. Let us begin with two definitions that will simplify our discussion.

Definition 7.1 (Fiber Point)**.** *A fiber is the inverse image of a point in the range. For convenience we will call the point of which we take a fiber of a fiber point.*

Definition 7.2 (Active Simplex)**.** *We call a simplex active when its intersection with a fiber is not empty.*

Next we will investigate the behaviour of a fiber within a single tetrahedron. A fiber can be expressed as the intersection of the isosurfaces of the component functions $(f, g)^{-1}(u, v) = f^{-1}(u) \cap g^{-1}(v)$. In a tetrahedral mesh we know the intersection of an isosurface and a tetrahedron is either empty, a single point, or a planar segment (the intersection of a plane and a tetrahedron) Treece *et al.* (1999b). Therefore, the intersection of a fiber and a tetrahedron is either empty, a single point or a line segment (see Figure 7.1). Furthermore all fiber line segments in a tetrahedron are parallel to one another. This can also be demonstrated using the "Generic Preimage Lemma" from Edelsbrunner *et al.* (2008c).

A direct consequence of this is that we can determine whether a simplex is active based entirely on the location of the fiber point in the range. If the image of a simplex contains the fiber point, then that simplex is active. Furthermore due to the linear interpolation of the PL map, the barycentric coordinates of the fiber point in relation to the image of an active simplex are the same as the barycentric coordinates of the point in relation to the simplex in the domain.

This allows us to compute a fiber in a tetrahedral mesh in a style similar to marching tetrahedra for isosurfaces Treece *et al.* (1999b). For each tetrahedron determine whether two of its proper faces are active by computing the barycentric coordinates of the fiber point with respect to the images of the simplices in the plane. If the simplices contain the fiber point use the same barycentric coordinates in the domain and draw the line segment between those two points of intersection. To compute the fiber in the whole mesh we apply this to every tetrahedron individually.

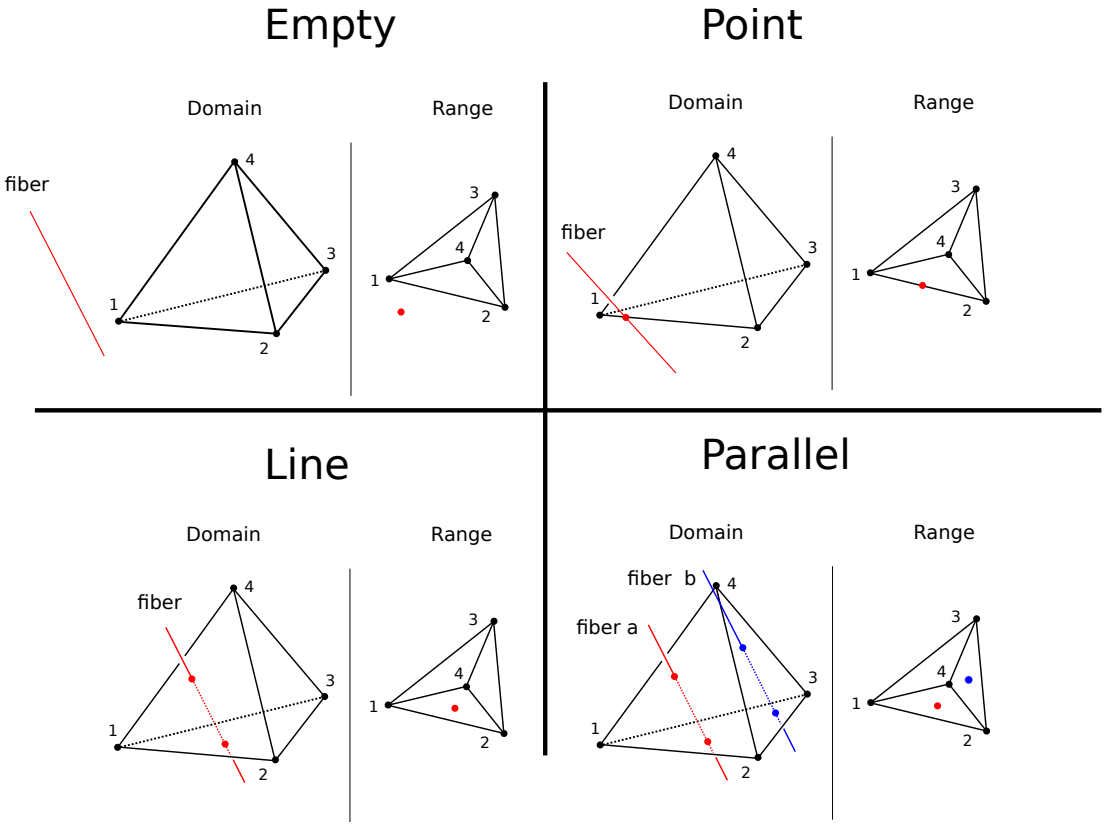


Figure 7.1: The possible ways in which a fiber can intersect a tetrahedron. We consider as our domain a single tetrahedron and its mapping onto the plane by a PL map. The PL map is defined on the vertices and linearly interpolated along the edges, triangles and tetrahedra. We construct the fiber by determining which faces of the tetrahedron contain the fiber point in the range, then connect them in the domain.

Based on this we will define two ways of thinking about fibers - geometric and combinatorial.

Definition 7.3 (Geometric Fiber)**.** *By a geometric fiber we mean the preimage of a point, or the fiber as computed with its corresponding geometry.*

Definition 7.4 (Combinatorial Fiber)**.** *By a combinatorial fiber we mean a graph where the vertices are the active faces for a fiber and an edge connects two vertices if their corresponding simplices are faces of the same tetrahedron.*

We make the difference between the two because in our discussion we often won't be interested in the exact geometry of the fiber. We will only be interested in the number of connected components of a fiber and how that changes, which we can obtain from the combinatorial description.

So far we have shown the behaviour of a fiber in a single tetrahedron and how to compute the fiber geometrically and combinatorially. The next step is to build on that to determine how a fiber behaves in adjacent tetrahedra. No topological change of the connectivity of the fiber can happen outside the edges and vertices of the mesh Edelsbrunner & Harer (2002b). Thus, when two tetrahedra meet at a triangle and the fiber point is outside the image of the 1-skeleton of the mesh the fiber in the two tetrahedra consists of the two line segments meeting at a point on the shared face 7.2.

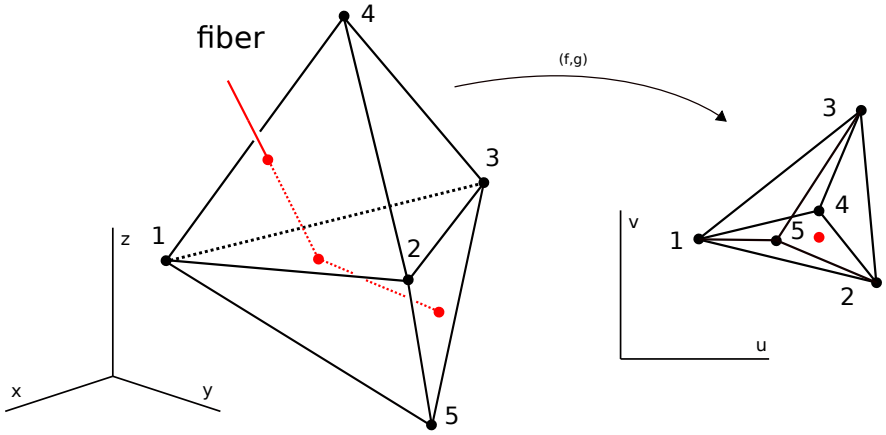


Figure 7.2: An example of when a fiber passes through the face of one tetrahedron onto another. The fiber is regular - locally a line segment.

There are two types of points in the domain and in fibers - regular and singular.

Definition 7.5 (Regular and Singular Points)**.** *A regular point is one where if we take a small neighbourhood of the fiber going through that point, the intersection of the fiber and the small neighbourhood is topologically equivalent to a line segment. Otherwise the point is called singular.*

Definition 7.6 (Regular and Singular Fibers)**.** *If all points on a fiber are regular, the fiber is called regular Otherwise it is called a singular fiber.*

The cases where topological change in the connectivity of the fiber happens is at the vertices and edges of the mesh. We will examine those two cases in the following two sections.

7.2 Local Edge Fiber Classification

So far we have looked at what happens to a fiber in a single tetrahedron and in two tetrahedra sharing a triangle. The next case is when multiple tetrahedra share an edge. This is where we can observe a topological change in the fiber, or the number of fiber components. In this section we will give a combinatorial description of all possible topological changes of a fiber near points on an edge of the mesh.

We can classify edges into a number of distinct types based on the behaviour of the fibers in a small neighbourhood around the edge. To demonstrate that we will take a walk around a point on the edge as discussed in Section 6.2. Before classifying the local behaviour of the fibers we will first describe the local neighbourhood of an edge in the domain and in the range. Those are the domain and range arrangement of the edge (recall Section 6.3).

7.2.1 Domain and Range Neighborhood Structure

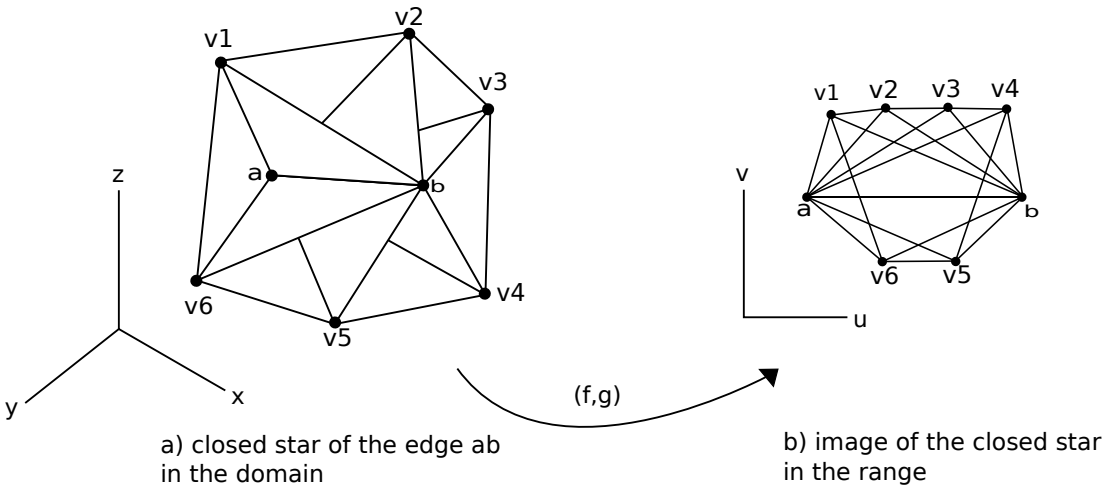


Figure 7.3: The neighbourhood of an edge in domain (left) and in the range (right). The link of the edge in the domain is the circle consisting of the points $v_1, \ldots, v_6$, and the closed star consists of all tetrahedra is of the form $(a, b, v_i, v_{(i+1)})$ for all $i \in 1, \ldots, 6$. In the range we've aranged the edge ab to be horizontal and all other vertices to be above or below it. Their exact location does not matter, only whether they are above or below the edge.

Given an edge ab of the mesh in the domain, by the neighbourhood of ab we mean the star of ab. In this work we consider the star of an edge to be closed - to contain all its faces. The star of an edge consists of at least three tetrahedra, all arranged around the edge as shown in Figure 7.3. The link of ab consists of a triangulation of a circle based on the vertices adjacent to ab. We will label those vertices as $v_1, \ldots, v_n$. There

is a topologically unique triangulation of a circle on n vertices so the structure of the star of an edge is entirely described by that single number n.

The neighbourhood of ab in the range is described by the image of the star of ab. From the work on Jacobi edges Edelsbrunner & Harer (2002b) we know that we can determine whether an edge is Jacobi entirely based on the images of the vertices $(f,g)(v_1),\ldots,(f,g)(v_n)$ and how they are arranged around the line defined by the image of the edge $(f,g)(ab)$. Geometrically there are infinitely many ways in which they can be mapped onto the plane. However Edelsbrunner & Harer (2002b) demonstrates that an edge can be classified as either a Jacobi or a regular edge, solely based on the connectivity its upper and lower link (recall Section 6.2).

Note that we will sometimes omit using $(f,g)(a)$ when we mean the image of a simplex a in the range. We will just write the simplex a in the domain or in the range, or it will be understood from context.

7.2.2 Choosing a Walk

Next we will describe how we choose the fiber point on the edge and the walk around it. It does not matter which which fiber point we pick on the edge, we will see later that this will not change the local behaviour of the fibers. For any point on the edge we pick the radius of the walk to be small enough so that it does not intersect any of the images of the edges of the star. If the point happens to be on the point of intersection of the central edge (ab) and another edge, that intersection is unavoidable, but does not change our local classification.

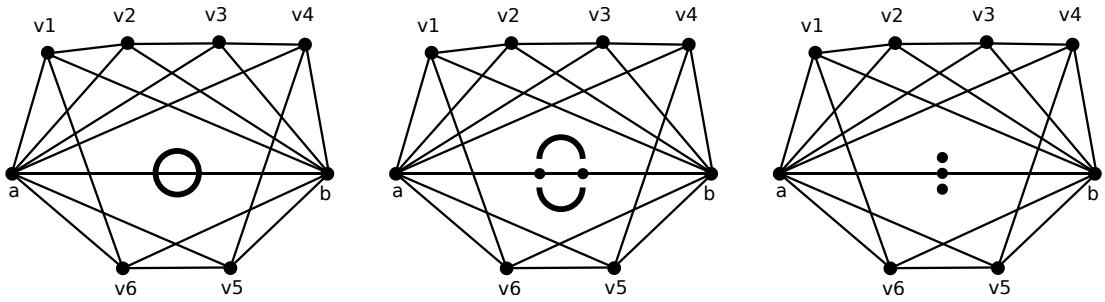
We can split the circle of the walk into three regions, each region having fibers with uniform topology Figure 7.4. The first region is the open half circle above the edge, the second region is the half circle in the region below the edge. Both regions are regular and therefore the fiber connectivity does not change for all points on the walk in them. The last region consists of the two points on the edge, those are singular, but still have the same topology.

7.2.3 Local Edge Fiber Classification

In order to determine the local fiber behaviour we need three fibers, two regular - one above and one below the edge, and one singular, at the edge. In order to compute the combinatorial fibers in the three regions we need to determine which triangles are active. We will group the triangles into two types - inner and outer triangles.

Definition 7.1 (Inner and Outer Triangles)**.** *In the star of an edge, an inner triangle is a face that is shared by two of the tetrahedra in that star. Every tetrahedron in the star has two inner triangles. All other triangles are called outer triangles.*

An inner triangle is a common face between two of the tetrahedra of the star of the edge. An outer triangle is a triangle on the boundary of the star of the edge. It



(a) A walk around a point on the edge ab.

(b) Regions where fibers have the same topology.

(c) The three fibers we need to describe the walk.

Figure 7.4: The fibers we need to talk to fully describe a walk around a point on an edge of the input mesh.

connects the star to the rest of the mesh. For example in Figure 7.3 all triangles abv_i for $i \in [1, 6]$ are inner triangles and all triangles av_iv_{i+1} are outer triangles.

We draw the distinction between the two types of triangles because is allows us to describe the fibers in the three regions around an edge. If a fiber passes through an inner triangle that means that it remains in the star, if a fiber passes through an outer triangle, that means it leaves the star and escapes into the rest of the mesh.

We give an example of computing a fiber above the edge ab in Figure 7.5. To compute the fiber we step through each of the tetrahedra in the closed star of ab one at a time (see Section 7.1). Each tetrahedron is either active or nonactive based on whether a fiber passes through that tetrahedron or not. Using the same method we can compute the fiber when the fiber point is on the edge or below the edge in Figure 7.6.

Next we are going to further classify each tetrahedron as either exit or nonexit based on the behaviour of the fiber inside that tetrahedron.

Definition 7.2 (Exit Tetrahedron)**.** *An exit tetrahedron is one where a fiber passes through one inner triangle and one outer triangle. Otherwise the tetrahedron is non exit.*

In non-exit tetrahedra the fiber comes in from an adjacent tetrahedron via an inner face and then exits to another adjacent tetrahedron via the other inner face. We can recognize these tetrahedra based on their image in the range. Both of the link vertices of a non-exit tetrahedron are on the same side of the line defined by ab. These are for example the tetrahedra abv_1v_2, abv_2v_3, abv_3v_4 and abv_5v_6 in Figure 7.5.

In exit tetrahedra the fiber comes through one of the inner faces and exits through one of the outer faces. The link vertices of an exit tetrahedron are on different sides of the line defined by ab. These are for example the tetrahedra abv_1v_6 and abv_4v_5 (Figure 7.6 and Figure 7.5). At the edge the fiber is entirely contained in the exit

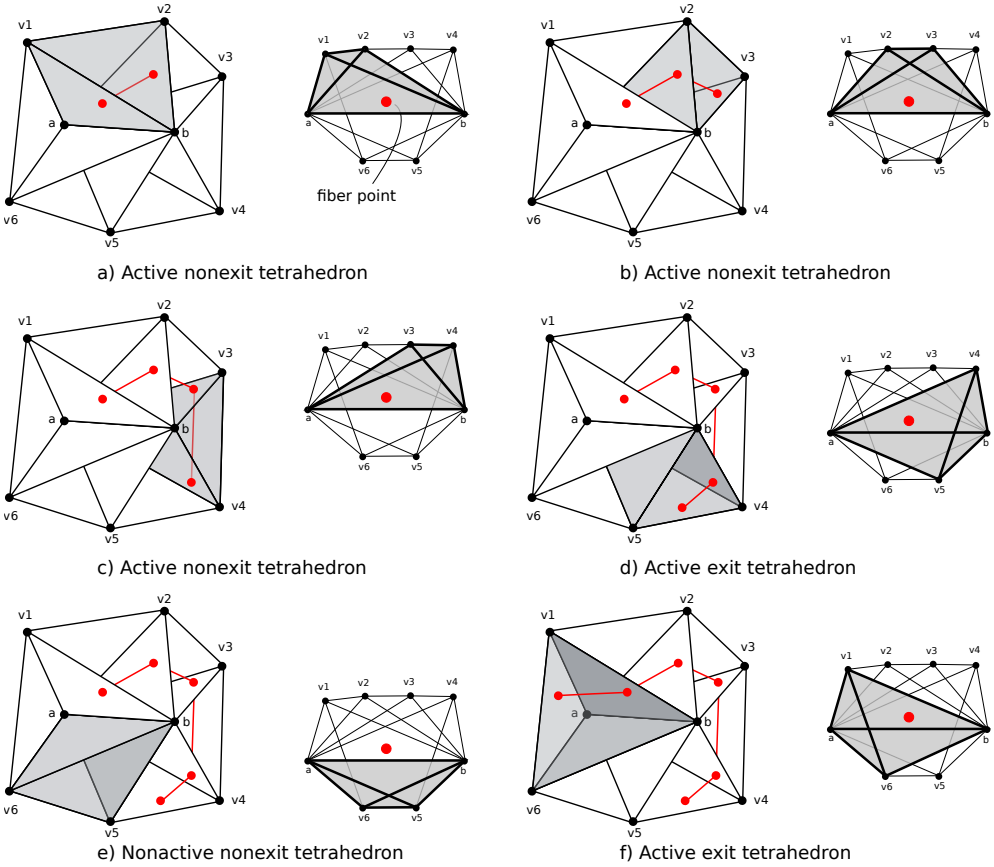
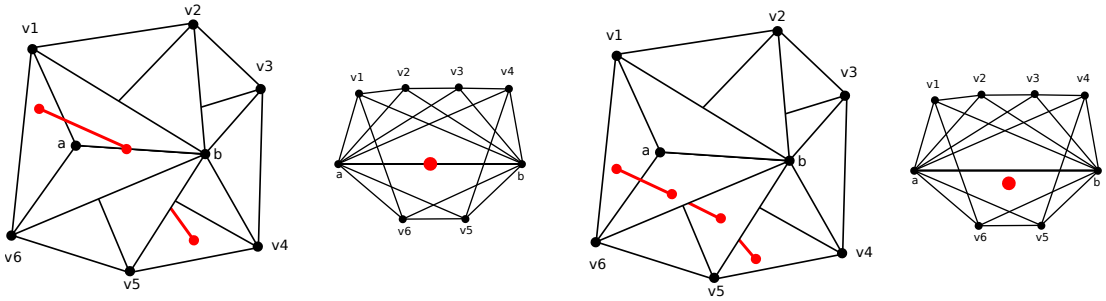


Figure 7.5: Computing a fiber in the star of an edge one tetrahedron at a time. The tetraheron shaded in gray (in the domain and in the range) is the current one being examined. In exit tetraherons the image of one vertex is above the adge ab, the other is below. Exit tetrahedra are always active, while nonexit tetrahedra are only active either above or below the edge.

tetrahedra (Figure 7.6 a)). Below the edge the same tetrahedra are exit tetrahedra, however the nonexit tetrahedra that were active in the fiber above the edge are no longer active (Figure 7.6 b)) and vice versa. Furthermore the nonexit tetrahedra that were not active in the fiber above the edge are active below the edge. In particular the only active nonexit tetrahedron below the edge is abv_5v_6.

Knowing about the two types of tetrahedra in the star of an edge we can now work on determining the types of Jacobi edges. As we mentioned before this will involve us taking three combinatorial fibers - above, at and below the edge. However we are not going to pick a specific fiber point for those combinatorial fibers. This is because all fiber points close enough and above the edge are contained in exactly the same images of triangles, and they all have the same combinatorial fibers. The only thing



a) Fiber at the edge

b) Fiber bellow the edge

Figure 7.6: The fibers at the edge and below the edge. The fiber at the edge connects the exit points directly to the edge, while passing only through the exit tetrahedra. The active tetrahedra of the fiber below the edge are the exit tetrahedra and all nonexit tetrahedra which were not active in the fiber above the edge. The only active non-exit tetrahedron below the edge is abv_5v_6.

that can change is that by moving the fiber point along the edge the fiber may move from one outer triangle of an exit tetrahedron into another. That does not change the combinatorial fiber topologically. This holds for all points close enough and below the edge and all points at the edge.

Notice that for non-exit tetrahedra if we pick a fiber point above (assume that the images of both link vertices are above) that is close enough, it will always intersect all interior triangles. If we also pick it close enough it will not intersect the exterior triangles. In an exit tetrahedron if we pick a close enough fiber point anywhere along the edge it will always intersect exactly one interior and one exterior triangle. We will call the point of intersection with an outer triangle an exit point.

Definition 7.3 (Exit Point)**.** *We call the point in the exit tetrahedron where the fiber exits the star of the edge an exit point.*

The edge we have considered so far is a regular edge because the fiber does not change its connectivity as it passes through the edge. In the star of a regular edge there are only two exit tetrahedra and therefore only one fiber component. We see fibers exhibiting singular behaviour when there are zero or more than two exit tetrahedra in the star of an edge. Next we will investigate the behaviour of the fibers based on the number of the exit tetrahedra. This will give us a full classification of the behaviour of singular fibers in the star of an edge.

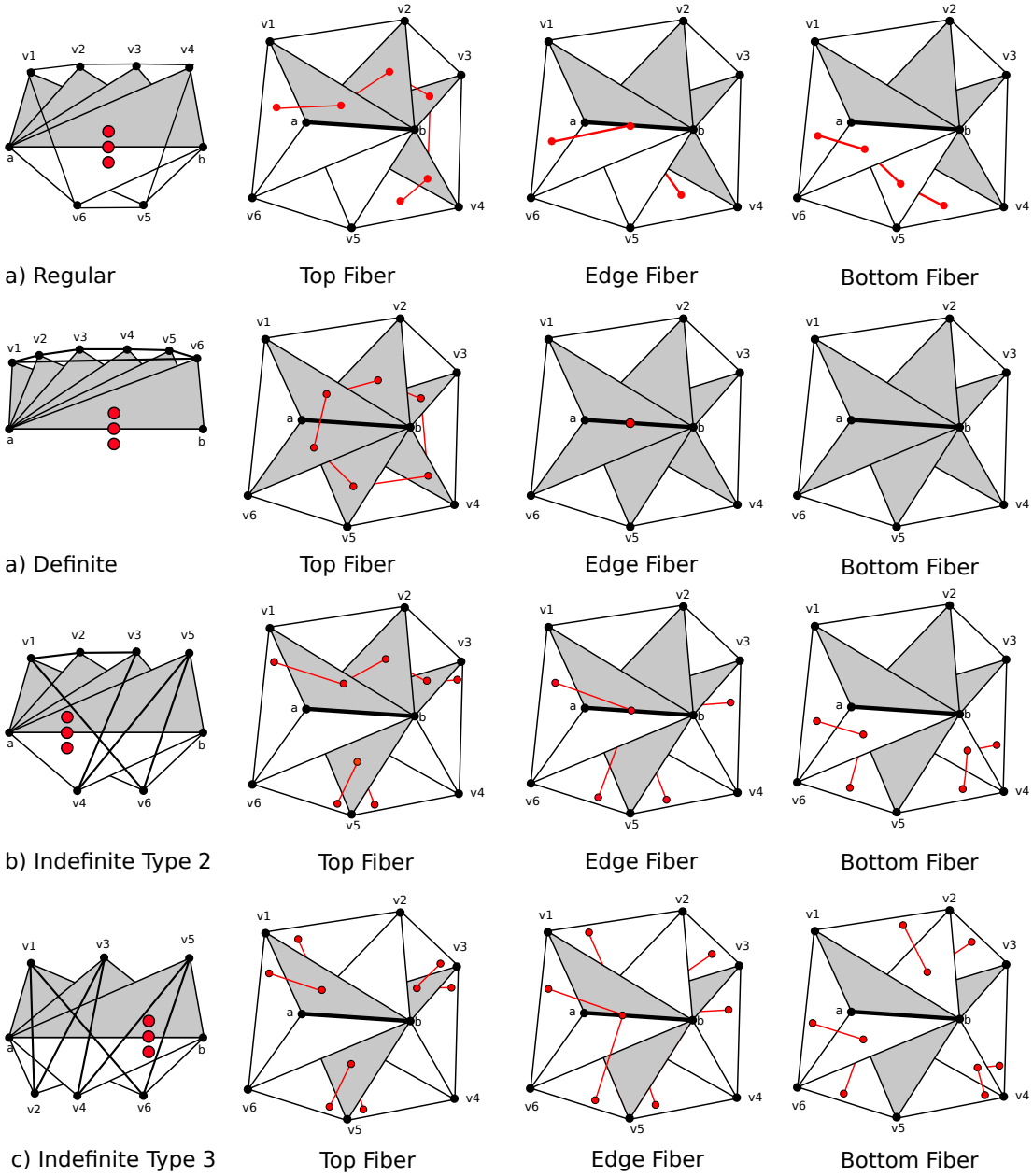


Figure 7.7: These are all possible Jacobi edge types for an edge of degree up to six. We have shown the transition of the fiber components as they move from above to at to below the edge.

Lemma 7.1 (Definite Edge)**.** *If the star of an edge has zero exit tetrahedra, then the fiber above the edge is a triangulation of a circle and it is entirely contained in the star. The fiber at the edge is a single point and the fiber below the edge is empty.*

Proof. If there are no exit tetrahedra then the images of all vertices on the link are on the same side of the line defined by $(f, g)(ab)$. Therefore the fiber passes through all inner faces. The fiber is a triangulation of a circle and it is entirely contained in the star. The fiber at the edge is single point and the one below is empty. The edge is then called a definite edge and you can see an example of this in Figure 7.7 a). □

In the other case there are more than two exit tetrahedra in the star of the edge. Let us first label all vertices on the link $v_1, \ldots, v_n$ as they appear clockwise on the link, and label all tetrahedra $T_1, \ldots, T_n$ as they appear clockwise as well. Tetrahedron T_i is made up of the vertices abv_iv_{i+1}. We will consider the list of vertices and tetrahedra to be circular, that is when we write T_{i+1} we mean T_1 when $i = n + 1$ and when we write T_{i-1} we mean T_n when $i = 1$. We are now going to use this new notation to show some important properties about exit tetrahedra.

Lemma 7.2 (Exit Point Parity)**.** *There is an even number of exit tetrahedra in the star of an edge.*

Proof. Consider the case of a definite edge where there are no exit tetrahedra. Assume without loss of generality that the images of all vertices are above the line defined by $(f, g)(ab)$. If we move the image of one vertex, say v_i below then the two edges $v_{i-1}v_i$ and v_iv_{i+1} now cross the line defined by $(f, g)(ab)$. Therefore the tetrahedra T_{i-1} and T_i are now exit tetrahedra. If we move any vertex adjacent to v_i below, the number of edges crossing the line (and therefore the number of exit tetrahedra) stays the same. If we move any vertex non-adjacent to v_i below the line then there are four edges crossing the line and therefore four exit tetrahedra. Inductively the rest follows. □

The way we describe fibers combinatorially requires us that we list all active triangles in the star and connect them together. However we can simplify this by connecting pairs of exit points directly.

Definition 7.7 (Exit Point Pairs)**.** *When a fiber connects two exit points in the star of an edge we say that the exit points are paired.*

Exit point pairs are a concise representation of combinatorial fibers in the star of an edge. Instead of listing all active triangles we consider a fiber just by the exit point it pairs. The following lemma tells us how exactly exit points pair with one another and how that pairing changes on the other side of the Jacobi edge. This is crucial for understanding the local behaviour of fibers.

Lemma 7.3 (Exit Point Pair Swap)**.** *Let $T_{k_1}, T_{k_1}, \ldots, T_{k_m}$ be the circular list of exit tetrahedra. Then a regular fiber above either pairs T_{k_i} with the previous tetrahedron $T_{k_{i-1}}$ or the next tetrahedron $T_{k_{i+1}}$. Furthermore if the tetrahedron T_{k_i} pairs with the*

previous one, then in the regular fiber below the edge the pairs flip and it pairs with the next one. Analogously if the tetrahedron T_{k_i} pairs with the next one, then in the regular fiber below the edge the pairs flip and it pairs with the previous one. This holds for all tetrahedra with the same index parity (odd or even).

Proof. Let us examine the fiber that enters the star at the exit point in T_{k_1}. Suppose without loss of generality that we are considering the regular fiber above the edge and that the fiber goes on into the next tetrahedron in the circular list (and not the previous). If the next tetrahedron is a non-exit tetrahedron then the fiber moves on to the one after that. This goes on until the fiber reaches an exit tetrahedron where the fiber exits the star. Since there is a only one fiber segment in every active tetrahedron there is nowhere else the fiber can go. Therefore the exit point p_{k_1} from tetrahedron T_{k_1} pairs with the exit point p_{k_2} from tetrahedron T_{k_2}.

All nonexit tetrahedra from T_{k_2} to T_{k_3} are inactive because the images of both of their inner triangles are below the edge. Therefore by the same reasoning the fiber that comes in the tetrahedron T_{k_3} pairs it with the next tetrahedron T_{k_4}. The same holds for all following tetrahedra.

The pairs of exit points then are $\{(p_1, p_2), (p_3, p_4), \dots (p_{m-1}, p_m)\}$. When we take the regular fiber below the edge all active non exit tetrahedra become inactive and vice versa so the exit points flip and become $\{(p_{m-1}, p_1), (p_2, p_3), \dots (p_{m-2}, p_{m-1})\}$

□

The Exit Point Pair Swap lemma completely describes the behaviour of local fiber components when the number of exit tetrahedra is greater than two. In Figure 7.7 we have shown the three types of Jacobi edges that can happen in meshes with edges of degree up to six. The first case is the definite edge, where there are no exit tetrahedra and hence the fiber above the edge is a circle. The fiber then shrinks to a point at the edge as it disappears below the edge. This is analogous to local minima and maxima in Reeb graphs where contours appear and disappear. This case is characterised by the fact that all link vertices are on the same of the line defined by ab.

The next possible case is the indefinite edge of type two, where we have two fiber components in the regular fibers. Above the edge one of the fibers pairs the tetrahedron v_1v_6ab with the tetrahedron v_2v_3ab since the images of the triangles abv_1, abv_2 and abv_3 are all above the line defined by the image of the edge ab. Similarly the tetrahedron v_5v_6ab is paired with v_4v_5ab since the image of the triangle abv_5 is below the line defined by the image of the edge ab. Below the edge the pairs swap as per Lemma 7.3 and v_1v_6ab pairs with v_5v_6ab and v_2v_3ab pairs with v_4v_5ab.

For the case of the indefinite edge of type three we have three fiber components in the regular fibers above and below. Same as before above the edge the fibers pair the exit tetrahedra via the inner triangles which are above the edge. At the edge we have

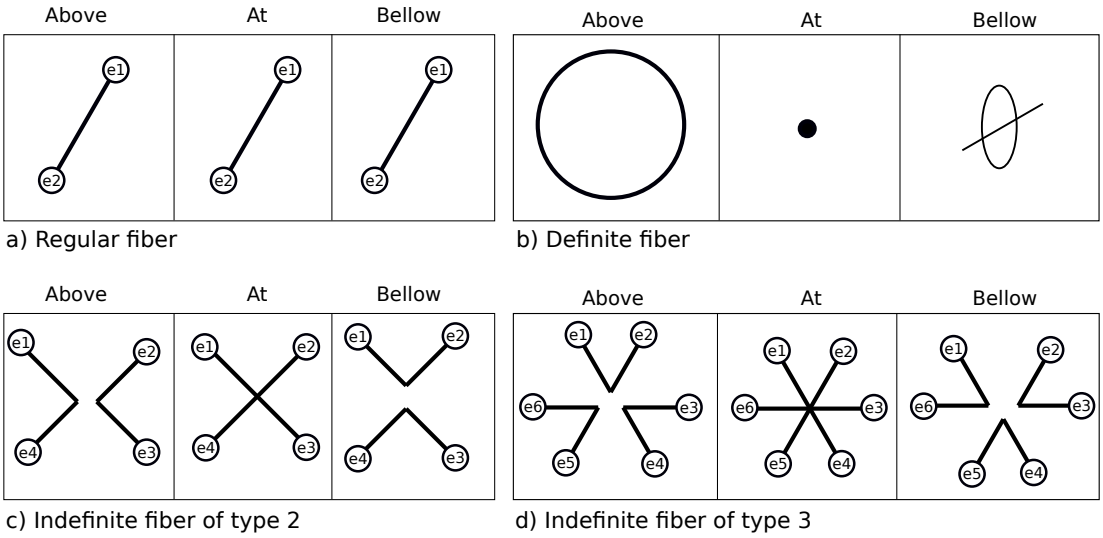


Figure 7.8: In every subfigure we have shown a simplified view of three fibers, one above, one at and one below a Jacobi edge. We classify the edge based on the behaviour of the fiber as it crosses the edge - it's either a regular, a definite or indefinite edge. Indefinite edges can be of type $2, 3, 4, \ldots$, but we have only shown 2 and 3 here, the rest are analogous with more fiber components and exit points (see Lemma 7.3).

a singular fiber, where all fiber components meet. Then they split again as the exit point pairs swap.

It's precisely the swap of exit points when we cross an indefinite edge that enables the topological change in connectivity. When we consider the connections of the fiber outside of the star of the edge, in the rest of the mesh, this swap may change the number of fiber components. We will describe this global process in more detail in Section 8.1. First we will describe the local behaviour of fibers around vertices of the mesh.

7.3 Local Vertex Fiber Classification

In this section we will use what we've learned so far about edge neighbourhoods to describe vertex neighbourhoods. Vertex neighbourhoods are more difficult to describe because multiple Jacobi edges can meet at a vertex. For example all types of edges such as regular, definite and indefinite can all meet at a vertex in various combinations.

In order to describe local vertex neighbourhoods we will use the same method as for the edges (recall Section 6.3). First we describe the structure of the star and the link of a vertex in the domain (the domain arrangement). Then we describe the structure of the image of the star of the vertex in the range (the range arrangement).

The key contribution of this section is to describe the range arrangement via a discrete combinatorial structure we call a dihedral top permutation (DTP). By gener-

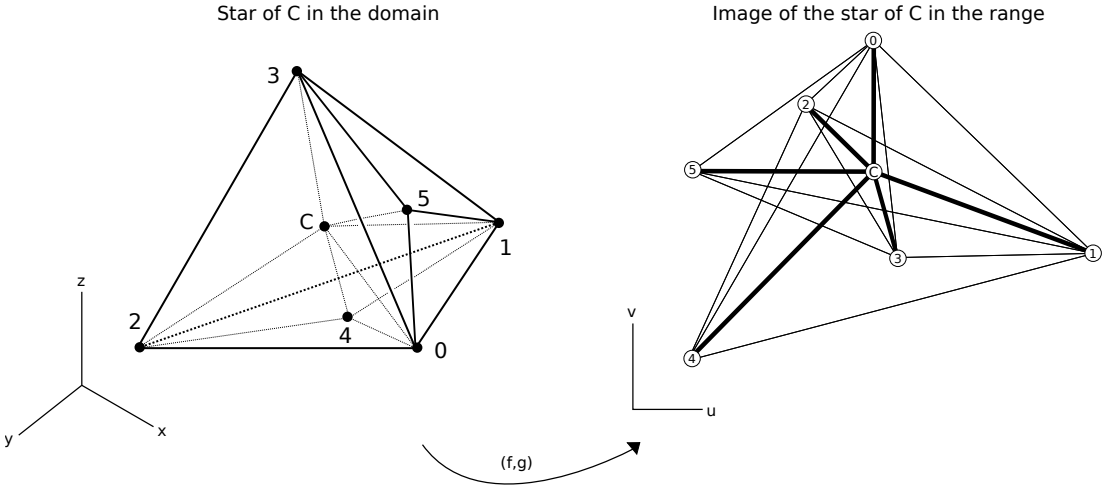


Figure 7.9: The structure of the star of a vertex C is the following. The link of C is a triangulation of a sphere. The star of C consists of tetrahedra whose base is a triangle on the link. As per our assumptions (see Section 6.4) the images of the vertices in the range are arranged so that no two vertices are on a line passing through C.

ating all DTPs we exhaust all possible range arrangements and we can thus generate all possible Reeb space neighbourhoods.

7.3.1 Structure of the Vertex Neighborhood in the Domain

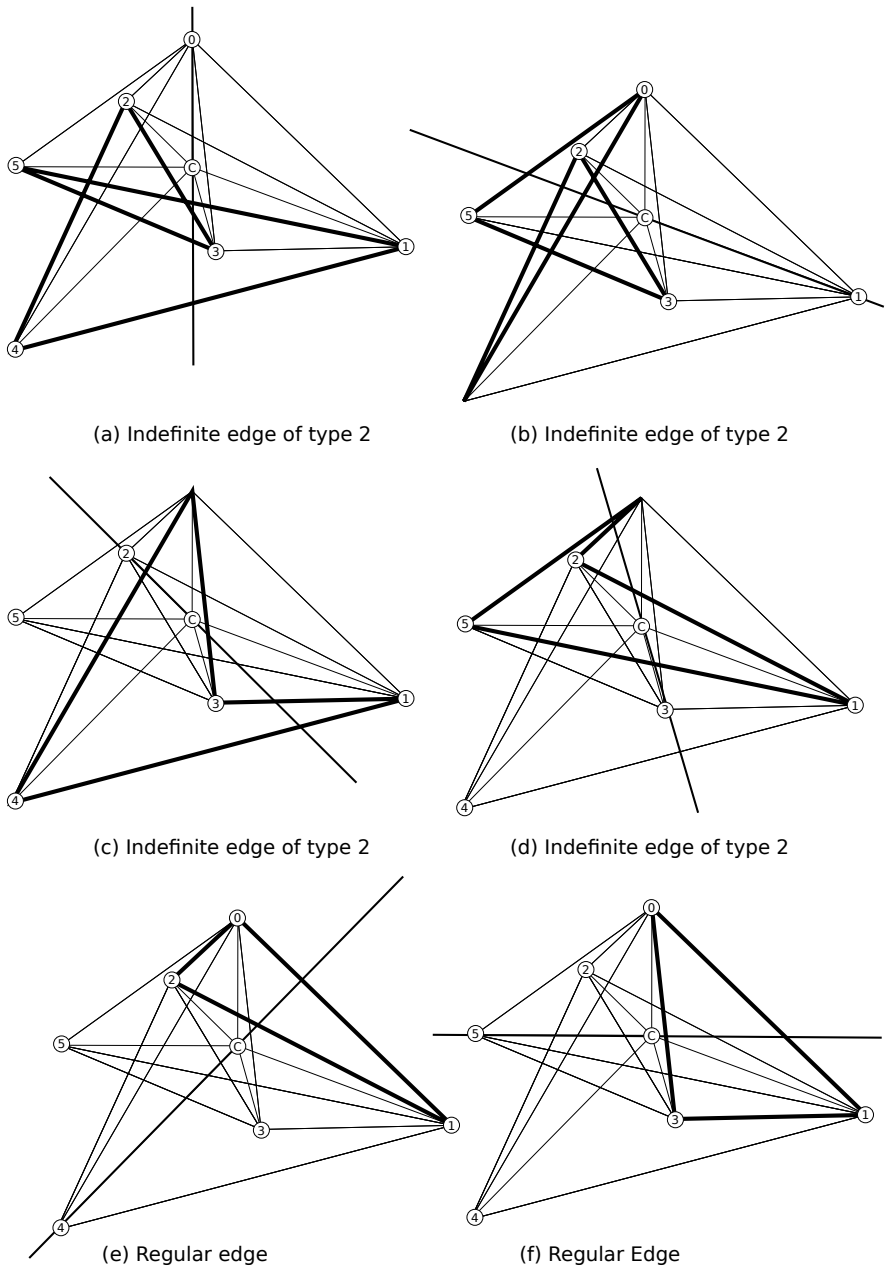
In a combinatorial 3-manifold without boundary the link of a vertex C is a triangulation of the sphere. We will call the vertex C the central vertex and the other vertices link vertices. The star of C consists of all tetrahedra of the form CT where T ranges over all triangles in the link (see Figure 7.9). The product CT referes to the cone operation of taking all line segments between C and T.

This is already a significant departure from the case of edge neighbourhoods where the link was a triangulation of a circle. While there is a unique triangulation of a circle on n vertices, there are many different triangulations of a 2-sphere on n vertices. By topologically different we mean graph isomorphic on the 1-skeleton of the triangulation.

In the star of a vertex we will differentiate between two types of simplices.

Definition 7.4 (Inner and Outer Simplices)**.** *We call the vertices which are entirely contained in the link of a vertex outer simplices. The other simplices in the star of a vertex will be referred to as inner.*

Throughout the rest of this section we will consider the triangulation of a sphere on six vertices that is given in Figure 7.9. Note however that the methods we present will be able to work with any given triangulation of a sphere on any number of vertices.



(a) Indefinite edge of type 2

(b) Indefinite edge of type 2

(c) Indefinite edge of type 2

(d) Indefinite edge of type 2

(e) Regular edge

(f) Regular Edge

Figure 7.10: An example of determining the Jacobi type of the edges adjacent to the vertex C. For every edge Ci where $i \in \{0, 1, 2, 3, 4, 5\}$ we consider the line defined by the edge and the link of that edge. The number of edges from the link that cross the line determines the Jacobi type of the edge (see Subsection 7.2.3).

7.3.2 Structure of the Vertex Neighborhood in the Range

Geometrically, there are infinitely many ways in which the images of the link vertices can be arranged around the image of the central vertex. However we want a combinatorial description that reduces the number of cases to a discrete set. The question we are going to answer is which types of arrangements of the link vertices in the range give us equivalent Reeb space neighbourhoods and which give us potentially different ones.

The key to understanding the local structure of the image of the vertex neighbourhood is to consider all the edges adjacent to the central vertex. The star of each of the adjacent edges is contained in the star of the vertex and we can use it to examine the edge's Jacobi type. You can see an example of this in Figure 7.10 where we examine the Jacobi type of each edge adjacent to the central vertex from Figure 7.9.

We can define the Jacobi type of a vertex based on the arrangement of the Jacobi edges around it. The Jacobi type consists of an ordered list of the Jacobi types of the adjacent edges of the vertex. In describing the Jacobi type of a vertex we use 0 for definite edges and $n \geq 2$ for indefinite edge of type n. We omit the regular edges because they do not contribute to the Reeb space structure. The Jacobi type of the vertex from our running example (see Figure 7.9) is $[2, 2, 2, 2]$ since the vertex has four adjacent indefinite edges of type two.

The Jacobi type of a vertex exhibits rotational and mirror symmetry. A vertex with Jacobi type $[2, 0, 2, 3]$ is equivalent to another vertex with Jacobi type $[3, 2, 0, 2]$ via a single rotation or a single a mirror operation.

While the Jacobi type of a vertex is easy to describe and useful in initially classifying vertices, it is difficult to determine which vertex Jacobi types are realisable and which ones are not. By realisable we mean whether a mesh and a PL map exist that have that particular Jacobi type. Therefore we will assume a bottom up approach of computing all realisable Jacobi vertex types first.

Another way to describe the local structure of the vertex in the range is to list the vertices from the link of the central vertex which are above and below for each edge individually. From this representation we can directly compute the Jacobi types of all the edges. However this representation results in multiple lists of vertices that is cumbersome to work with because there is a lot of redundancy. Instead we propose an alternative representation called a dihedral top permutation (DTP).

In the term dihedral top permutation (see Figure 7.11) top means that some of the vertices have bars on top that signify which side of the line defined by their edge we encounter them as we do a clockwise scan. If the vertex is on the same side of the line as the edge it does not have a bar, if it is on the opposite side, it does have a bar. We use the term permutation because each vertex and its top counterpart appear exactly once and the order matters. We call it dihedral because it exhibits rotational and mirror symmetry. For example the DTP $[v_0, \bar{v_2}, v_1, \bar{v_0}, v_2, \bar{v_1}]$ is equivalent to the DTP $[v_1, \bar{v_2}, v_0, \bar{v_1}, v_2, \bar{v_0}]$ via two unit rotations and a mirror flip.

For convenience we will not always write out the full DTP, we can in fact only write half of it. Note that for example in the DTP $[v_1, \bar{v_2}, v_0, \bar{v_1}, v_2, \bar{v_0}]$ once we reach $\bar{v_1}$ which is the half turn point, the vertices repeat in inverse order and with opposite bars

(the ones that had a bar don't and vice versa). Therefore we can only write $[v_1, \bar{v_2}, v_0]$ and the rest of the DTP is implicit. The second is that out of all equivalent DTPs we can pick a unique representative by always starting out with v_0 (this excludes the rotational symmetry) and picking the DTP in which we see v_1 before $\bar{v_1}$ (this excludes the mirror symmetry).

We will demonstrate the first use of the DTP notation by showing how we can compute the Jacobi type of a vertex.

Lemma 7.4. *We can use the DTP of a vertex to compute the Jacobi type of that vertex.*

Proof. Suppose that we are interested in computing the Jacobi type of the edge cv_i. Starting at v_i in the DTP we move along the list until we reach $\bar{v_i}$. All vertices which we encounter with a bar are on one side of cv_i (above) and all vertices which we encounter without a bar are on the other side (below). Once we have the vertices above and below we can count the number of edges from the link of cv_i which cross the line defined by cv_i. This gives us the Jacobi type of edge cv_i. Computing the Jacobi type of all edges in order gives us the Jacobi type of the vertex. □

The key thing that we need to show now is that any two arrangements of the link vertices in the plane with the same DTP are equivalent. By equivalent we mean that they generate the same Reeb space neighbourhoods up to rotational and mirror. We can do so by showing the we can use a DTP to compute the combinatorial fibers in the star of the central vertex over a walk around it. Therefore any two arrangements with equivalent DTPs will generate equivalent combinatorial fibers in the regions of the walk and therefore produce equivalent Reeb space structures.

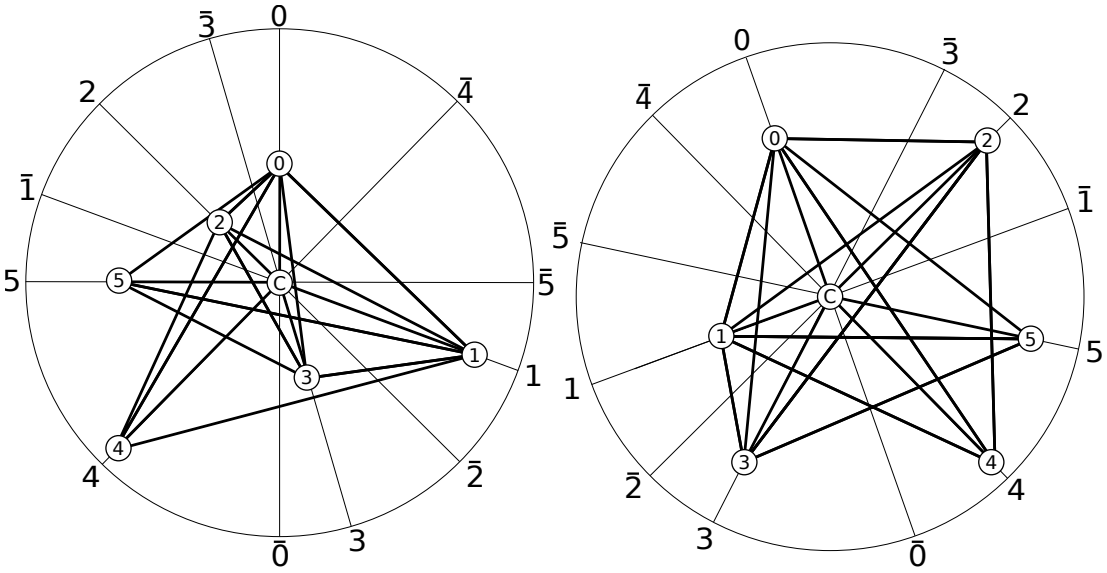
7.3.3 Walks Around Vertices

Given our assumptions (see Definition 6.1) we know that we can always pick a small enough walk (or circle) around the central vertex such that it only intersects inner edges. Since all the exit points of the vertex star are triangles on the link, those triangles do not contain central edges. Therefore over a small enough walk the exit points will remains constant in their triangles (even if they move around by a small amount).

We will segment the points on the walk around the central vertex into regions (see Figure 7.12). Jacobi regions represent adjacent Jacobi edges and regular regions represent the regular space between them. Over the walk around the central vertex we will compute a combinatorial fiber for each one of those regions. This will give us a full description of the inner fiber arrangement.

7.3.4 Local Fibers via Regular-like DTP Expressions

For computing the combinatorial fiber of a region on the walk around a vertex we need to know which the active triangles are for any point within that region. We won't have



(a) Example of a range arrangement with a dihedral top permutation (DTP) $v_0, \bar{v_4}, \bar{v_5}, v_1, \bar{v_2}, v_3$.

(b) A geometrically different arrangement of the link vertices with an equivalent DTP (rotated and mirrored).

Figure 7.11: Diagram of how to read the dihedral top permutation (DTP) of the image of the star of a vertex in the range. The DTP tells us how the vertices located on the link of the central vertex are arranged after they are mapped to the plane.

to do any geometric calculations to obtain the active triangles we can compute them entirely using the domain arrangement and the DTP (which is derived from the range arrangement).

We will determine whether a triangle is active by taking its vertices and matching them as a pattern onto the DTP. Since that is similar to how you would match a regular expression to a string we call this technique regular-like DTP expressions.

Regular-like DTP Expressions

In this section we will separate the triangles of the star of a vertex into three types. The outer triangles, the inner triangles for a Jacobi region and the inner triangles for a regular region. The outer triangles are significant because that's where the exit points are. The inner triangles are significant because they may contain exit points for the stars of individual edges. Those inner exits points then lead to exit points on outer triangles through the other tetrahedra in the star of the vertex.

First we will discuss detecting whether an outer triangle is active for any of the regular or Jacobi regions of the walk (Figure 7.13 a)). The outer triangles consists of three vertices from the link of the vertex. Suppose without loss of generality that those vertices are labeled as v_0, v_1, v_2 When we write that a DTP contains the subsequence

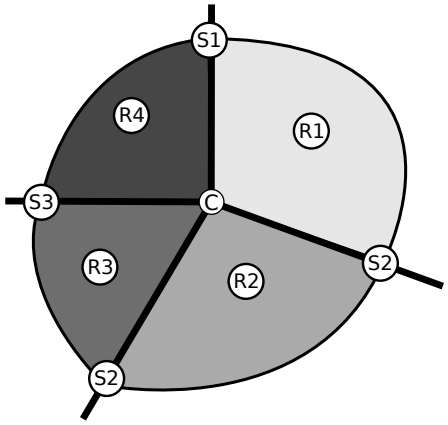


Figure 7.12: In a walk around a vertex there are multiple regions - one for every adjacent Jacobi edge and one for the region between two consecutive Jacobi edges. These are the regular and Jacobi regions of the DTP shown in Figure 7.9. In order to describe the walk it's enough to compute the combinatorial fiber for each one of the regular and Jacobi regions.

$(\ldots, v_0, \ldots, v_1, \ldots, v_2 \ldots)$ we will just write it as (v_0, v_1, v_2), but keep in mind they do not to be consecutive.

Lemma 7.5. *Given a triangle $v_0v_1v_2$ in the link of the central vertex c, the image of that triangle contains the image of the central vertex when the DTP of the arrangement contains the subsequence $(v_0, \bar{v_2}, v_1)$.*

Proof. Throughout this proof refer to Figure 7.13 a) for a diagram. Fix the positions of v_0 and v_2 in the plane. The line defined by the vector cv_0 separates the plane into two half-planes. Let us call those $P^+_{cv_0}$ which is the one clockwise of cv_0 and $P^-_{cv_0}$ which is counter clockwise of cv_0. In general for any two vectors ab and ac we have that if $c \in P^+_{ab}$ then $b \in P^-_{ac}$. In other words if c is counter clockwise of ab then b is clockwise of ac. Without loss of generality assume that $c \in P^+_{v_0v_1}$. Then if we show that $c \in P^+_{v_1v_2}$ and $c \in P^+_{v_2v_0}$ by the half-plane test c is in the triangle $v_0v_1v_2$. We have that $c \in P^+_{v_1v_2}$ whenever $v_2 \in P^-_{v_1c}$. This means that v_2 must be in the lower half-plane with respect to the line defined by v_1c. Furthermore $c \in P^+_{v_2v_0}$ whenever $v_2 \in P^-_{cv_0}$. This means that v_2 must be in the left half-plane of the line defined by cv_0. Therefore v must be in the lower left quadrant, so $\bar{v}$ must be in the top right quadrant, which corresponds to the DTP subsequence $(v_0, \bar{v_2}, v_1)$. □

The second technique is used to determine whether the image of an inner triangle has a non-empty intersection with the image of a particular edge (Figure 7.13 b)). Due

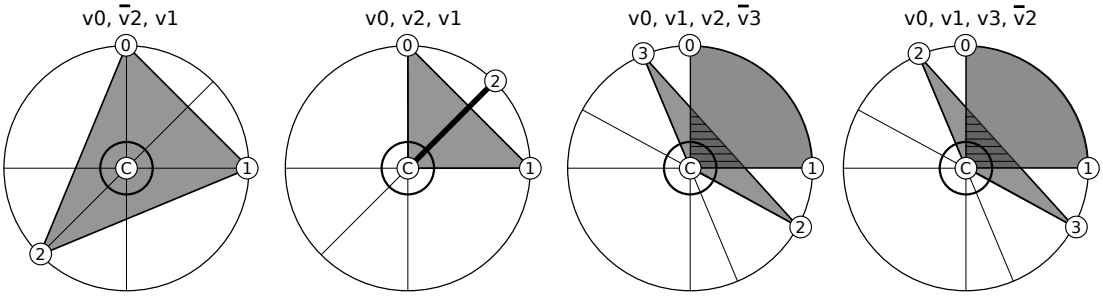


Figure 7.13: Recognizing whether particular points on the walk around c are in image of a triangle from the star of c. Above each image in the diagram is a subsequence of the DTP of the vertex, which we can match as a regular expression onto the DTP to determine whether the triangle is active. The first case is used to determine which triangles from the link of c are active. They will be active for every region on the walk, so we only need to compute them once. The second and third case are used to determine whether an inner triangle is active in a Jacobi or regular region respectively.

to how we have chosen our walk in Subsection 7.3.3 any such triangle must be active for that Jacobi region. The regular-like DTP expression we use to match this case is (v_0, v_2, v_1). We will omit the proof since as you can see in Figure 7.13 b) the only case when the edge cv_2 intersects the triangle v_0v_1c is when v_2 is in the upper right half-plane.

The final technique is used to determine whether a regular region on a walk has a non-empty intersection with an interior triangle (Figure 7.13 c)). Similarly to the previous two cases, if there is a non-empty intersection then we can always choose a walk small enough such that that triangle is active for all points in the regular region of the walk. In this case the regular region is defined by two vertices, say v_0 and v_1 so we match either of the expressions $(v_0, v_1, v_2, \bar{v_3})$ or $(v_0, v_1, v_3, \bar{v_2})$. Since we define regular regions to be regions between consequitive edges, we can assume that v_0 and v_1 are consequitive in the DTP. Therefore we reduce the proof of this case to the previous case, that of Figure 7.13 b).

Inner Fiber Arrangement

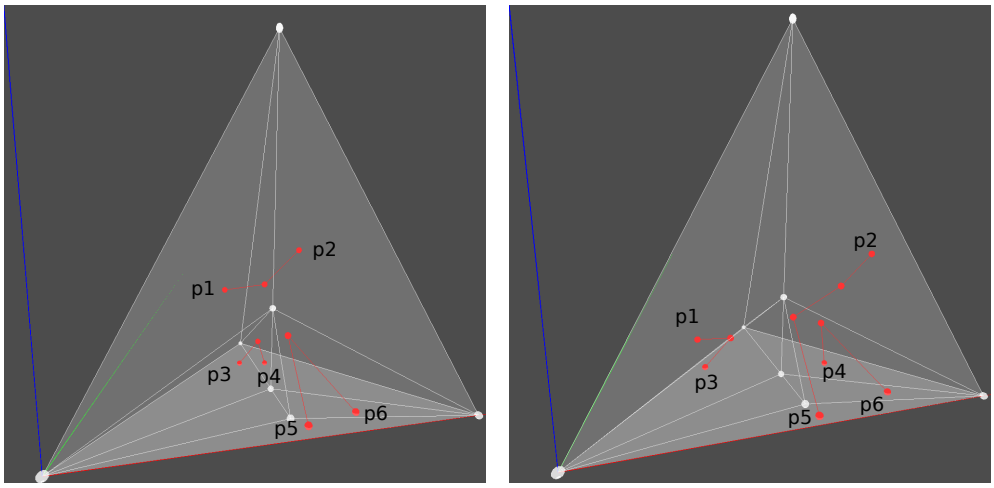
Given that we can compute the active triangles in each region of the walk using the DTP, this means that we can compute the combinatorial fibers in each region. Therefore any two geometric arrangements of the images of the link vertices in the range with the same DTP are indeed equivalent. This means that they have equivalent inner fiber arrangements (up to symmetry and relabeling).

We express the inner fiber arrangement as a list of exit point pairs (see Figure 7.14). Each pair represents a combinatorial fiber via its exit points on the link of the

central vertex. You can see an example of the geometric fibers of the regular regions in Figure 7.15. When the central vertex has adjacent definite edges some regular regions will have combinatorial fibers which are entirely contained in them (circles around the edge). We cannot express those as pairs of exit points so we add some auxiliary points p_i for each circle fiber and pair them with themselves (p_i, p_i).

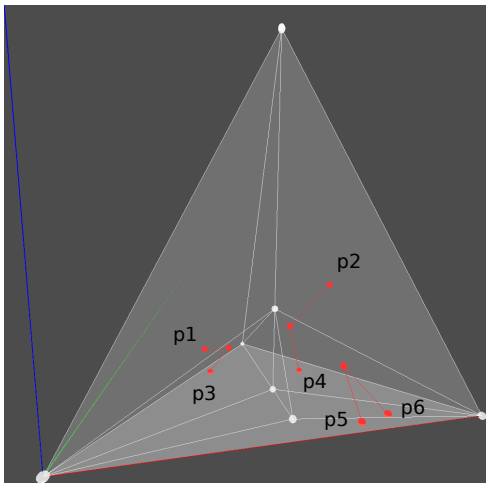
R1	S1	R2	S2	R3	S3	R4	S4
(p1, p2) (p3, p4) (p5, p6)	(p1, s) (p2, s) (p3, s) (p4, s) (p5, p6)	(p1, p4) (p2, p3) (p5, p6)	(p1, p2) (p3, s) (p4, s) (p5, s) (p6, s)	(p1, p2) (p3, p4) (p5, p6)	(p1, s) (p2, s) (p3, s) (p4, s) (p5, p6)	(p1, p4) (p2, p3) (p5, p6)	(p1, p2) (p3, s) (p4, s) (p5, s) (p6, s)

Figure 7.14: The local fiber arrangement diagram for the example from Figure 7.9. This table shows the local fiber pairings in all regular and Jacobi regions of the walk. In the Jacobi regions the pairs (p_i, s) means a singular fibber connecting the exit point to a point on the Jacobi edge which defines the Jacobi region.

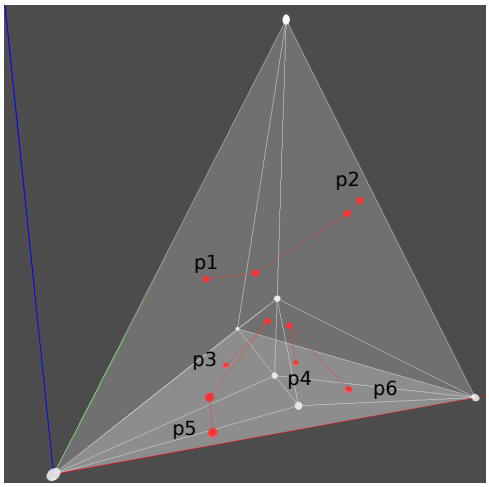


a) Fibers in Regular Region 1

b) Fibers in Regular Region 2



c) Fibers in Singular Region 3



d) Fibers in Singular Region 4

Figure 7.15: Showing the fibers in all four regular regions from Figure 7.9. The fiber connectivity gives us the exit point pairs in every region.

Chapter 8

Reeb Space Computation

8.1 Global Fiber Pairing for Local Reeb Space Neighborhoods

So far we have examined the local behaviour of fibers around a point that is on an edge or a vertex of the mesh. We categorized the possible ways in which fibers change their connectivity and described that with the local fiber arrangement. In this section we will consider how to extend the local behaviour of the fibers globally, so that we obtain their global connectivity. We will describe how the fibers connect outside the closed star of an edge or a vertex with a structure we call the outer fiber arrangement. Combining the outer fiber arrangement with the inner fiber arrangement gives us the global connectivity of fibers. This will allows us to infer the structure of the local Reeb space neighbourhood.

In this section we will work with some concrete examples to illustrate the process of obtaining the outer fiber connectivity and the local Reeb space neighbourhood. We will focus on the intuition behind the process and later use that to develop general algorithms for computing all possible local Reeb space neighbourhoods in Section 8.2.

8.1.1 Reeb Space Generating Diagrams

In the inner fiber arrangement each pair of exit points represents a fiber component inside the star in a particular region of the walk around the point. In order to study the structure of the local Reeb space, however we need to understand the change in global connectivity of the fibers. To understand this we need to know how the fiber components are connected in the rest of the mesh, outside the star.

From our assumptions we know that the input mesh is compact and it does not have a boundary and from basic topology we also know that the preimage of a point is a closed curve. Therefore when a fiber exits the closed star via one of the exit points, then it must travel around in the mesh and come back through one of the other exit

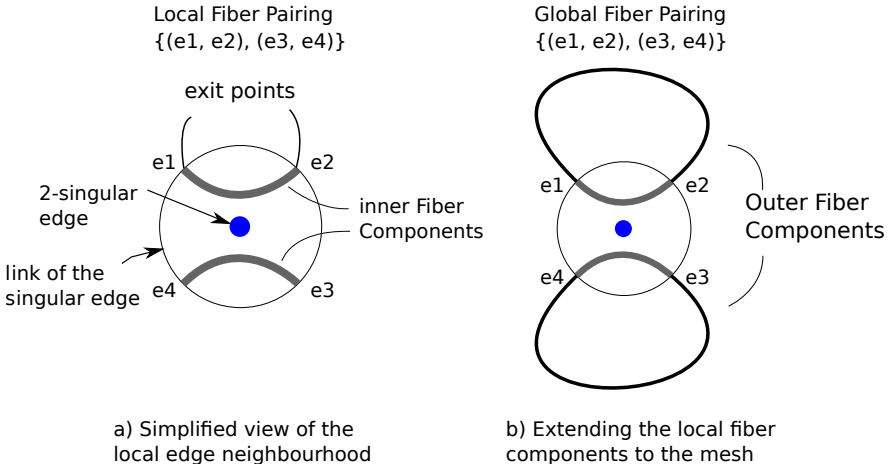


Figure 8.1: This diagram depicts the global connectivity of a fiber as a combination of the local and outer fiber pairings. On the left, is a simple visualisation of the star of an edge, where the edge is reduced to a point. On the right, the exit points are connected outside of the star in ther rest of the mesh.

points. This means that a list of pairs of exit points is enough to completely describe the global connectivity of fiber components outside of the star.

We combine the inner and outer fiber arrangements to get the global fiber components visually with a global fiber diagram diagram. You can see an example of this in Figure 8.1, where we present a stripped down visualisation of the components of a fiber as made up of its inner and outer fiber components.

8.1.2 Global Fibers of a Single Edge

We will now demonstrate how to combine the inner and outer fiber arrangements to the get the global connectivity of a fiber over a walk. We will do so by visualising the outer fiber as a combination of the inner and outer fiber pairings in each of the regular and Jacobi regions of the walk. We give an example in Figure 8.2. Both possible inner fiber pairings are given on either side of the edge (see Figure 7.8). Along the edge we have singular fibers where all exit points are connected to a point on the Jacobi edge. Outside the closed star we've selected a particular outer fiber pairing, which as we've discussed remains constant over the walk.

In the resulting diagram (Figure 8.2), we have three fiber components in the left, which merge at the edge into a single fiber component on the right hand side of the diagram. The merge happens because of the how the exit point pairs swap due ot the Exit Point Pair Swap Lemma (see Lemma 7.3). The resulting local Reeb space neighbourhood has one sheet on one side of the singular edge and three sheets on the other.

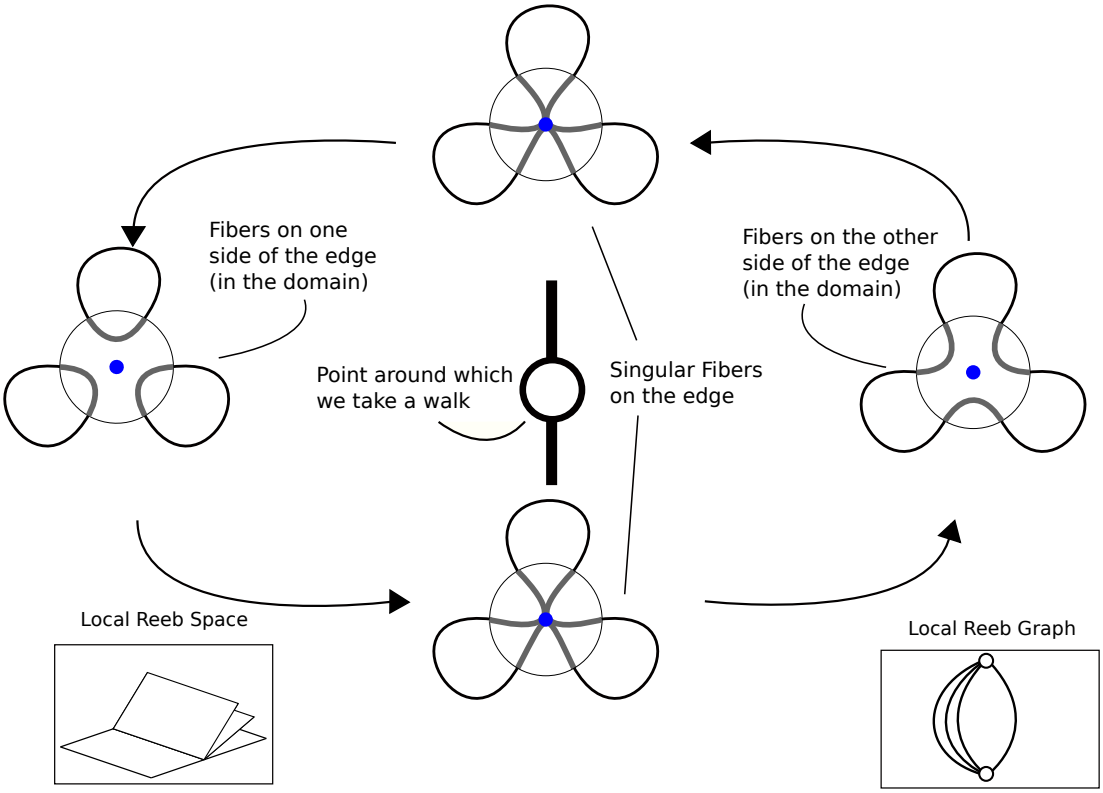


Figure 8.2: A walk around a point on a Jacobi edge in the range with outer fiber diagrams in each of the regions. On the bottom we've shown the Reeb space of the neighbourhood of the point and the Reeb graph of the walk.

8.1.3 Global Fibers of of Two Intersecting Edges

The global fiber could possibly intersect two singular points. This happens when the images of two edges intersect in the range and the fiber point is the point of intersection. Due to our assumptions (see Section 6.4) we know that no more than two edges can cross in the range. Therefore the fiber can intersect at most one other singular point on one other edge. Note that the fiber cannot intersect another vertex because that violates our other assumption.

When two edges intersect in the range the walk around the point of intersection will contain eight regions, instead of three (see Figure 8.3). There are four Jacobi regions which correspond to the edges, each appearing twice. There are also four regular regions in between the Jacobi edges. Over the course of the walk the inner fiber pairs of both stars flip, alternating, one at a time, as we cross each edge twice.

In the case of taking a walk around the point of intersection of two edges we have twice the number of exit points. In Figure 8.3 we give an example of a particular global pairing of the exit points of both stars. In all four regular regions we have all the

possible combinations of the exit point flips of both edges. For all Jacobi regions we have one edge being intersected by the fiber and the other one not. At the center point both edges are intersected by the fiber which has two singular points.

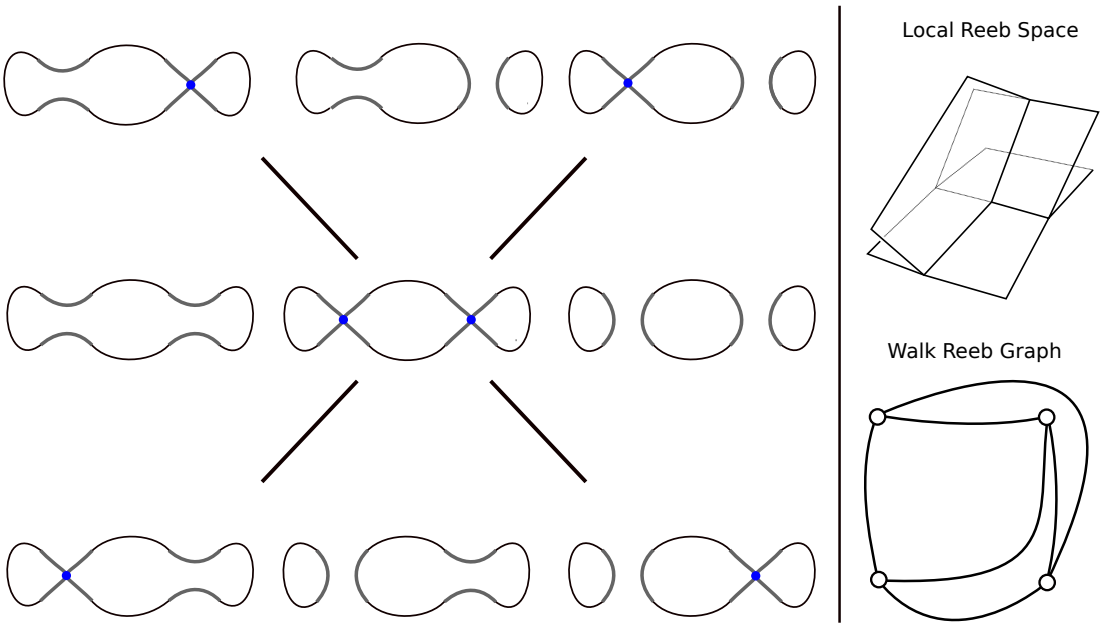


Figure 8.3: On the left we have the range, where the images of the two edges intersect. This divides the walk around the point of intersection into four regular regions and four Jacobi regions. The Jacobi regions are the Jacobi edges and the regular regions are the space between them. In each regular and Jacobi region in the range we have shown the global fiber (from the domain). Each of the fibers contains local fibers, inside the star of the edge and global fibers, the parts of the fiber outside the star. The singular fibers cross either one of the edges, or both (in the center). On the right we have shown the Reeb space neighbourhood of this case as well as the Reeb graph of the walk.

8.1.4 Global Fibers of a Vertex

When we examined the global fiber behaviour of edges we considered a special case, where the global fiber may have more than one singular point. This happens then the images of two edges intersect in the range. One of our initial assumptions (see Definition 6.1) was that no three vertices meet at an edge. Furthermore we have chosen our walk (see Subsection 7.3.3) to only intersect the images of the edges adjacent to the central vertex. Therefore the global fibers of vertex neighbourhoods contain one singular point.

In Figure 8.4 we have given an example of a Reeb space diagram for a particular global exit pairing for the vertex neighbourhood given in Figure 7.9. In the diagram we have given the combinatorial fibers of the four regular and four Jacobi regions. We have

also given the combinatorial fiber at the vertex in the center. In each combinatorial fiber we have arranged the exit points horizontally. The fiber components below the exit points represent the part of the fiber in the star of the vertex and the fiber components above the exit points represent the part of the fiber outside the star, in the rest of the mesh. At the Jacobi regions we have denoted the singular point in blue. On the right we have given the Reeb space of this example, as well as the Reeb graph of the walk.

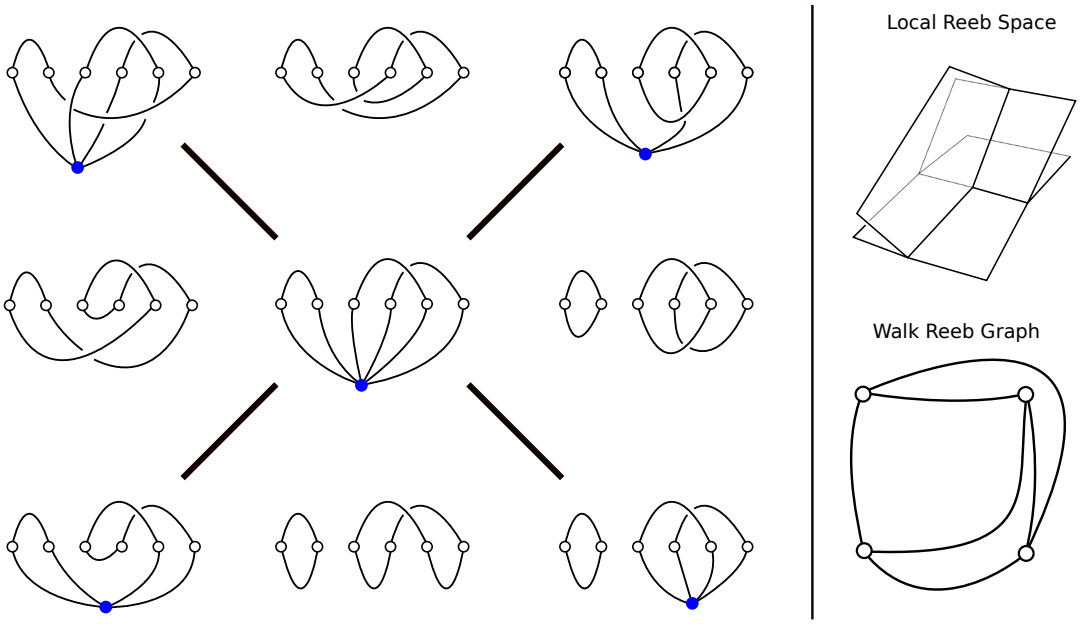


Figure 8.4: Showing the change in fiber connectivity over a walk around the central vertex from Figure 7.9. Each of the nine combinatorial fibers on the left hand side of the diagram consists of exit points, arranged horizontally; inner fibers in the star of the central vertex, arranged below the exit points; and global fibers arranged above the exit points. In the transition between the regions in the diagram the interior fibers change as they go though the singular points (shown in blue) on the Jacobi edges. Note that the central fiber is at the image of the central vertex. We have omitted labeling all the edges and exit points, since that is not significant for understanding this example. On the right hand side we have shown the Reeb space of the whole neighbourhood. In each region there is one sheet per fiber component. We have also shown the Reeb graph of the fibers on the walk.

8.2 Generating Reeb Space Neighborhoods

In Sections 7.2 and 7.3 we introduced the theory behind inner fiber connectivity and in Section 8.1 we introduced the idea behind global connectivity. Then in Section 8.1 we showed how we can combine the two to manually construct a local Reeb space neighbourhood. However we can only manually enumerate the possible local Reeb space neighbourhoods when the edge or vertex degree is small enough. Otherwise going through all possible arrangements manually quickly become prohibitive. In this section we will introduce algorithms to generate all possible Reeb Space neighbourhoods for meshes with bounded vertex and edge degree.

In order to generate Reeb space neighbourhoods we require the domain arrangement and the range arrangement to generate all possible inner fiber arrangements (see Section 6.3). Then we combine all possible inner fiber arrangements with all possible global fiber arrangements to obtain all possible Reeb space neighbourhoods.

8.2.1 Generating Outer Fiber Arrangements

To generate all possible global arrangements we first generate all pairs of exit points, then we select $n/2$ pairs such that every exit point is in exactly one pair and the order of the pairs in the selection does not matter. The total number of global arrangements on n exit points is:

$$\frac{\binom{n}{2} \times \binom{(n-2)}{2} \times \ldots \times \binom{2}{2}}{(n/2)!} = \frac{n!}{2^{n/2} \times (n/2)!} = \frac{n \times (n-1) \times \ldots \times n/2}{2^{n/2}}$$

.

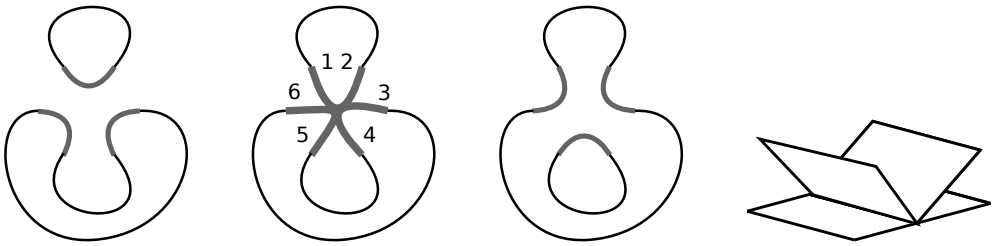
The total number of global pairings is quite large. Therefore we will consider some symmetry to reduce the total number of cases.

8.2.2 Dihedral Symmetry for Edges

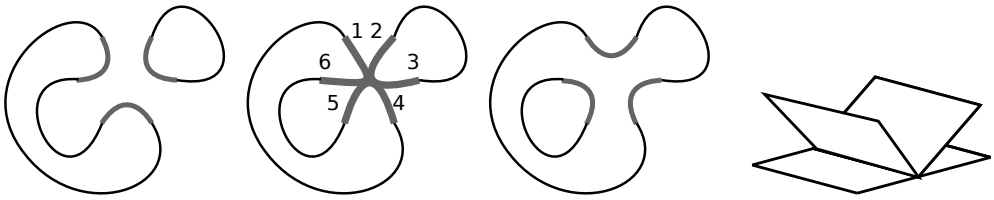
Consider Figure 8.5 where we have shown three distinct global pairings which are in fact equivalent. In the middle row we have shown the singular fiber along with the labeled exit point. All three cases can be expressed as a rotation or mirror of one another. More generally if we take the set of all global combinations we can define an equivalence relation based on rotational and mirror symmetry.

In the case of indefinite edges of type n we have $2n$ exit points points, so we take the Dihedral group of the hexagon D_{2n}. For the case of two intersecting edges of type n and m we consider the product of all their possible symmetries $D_{2n} \times D_{2m}$.

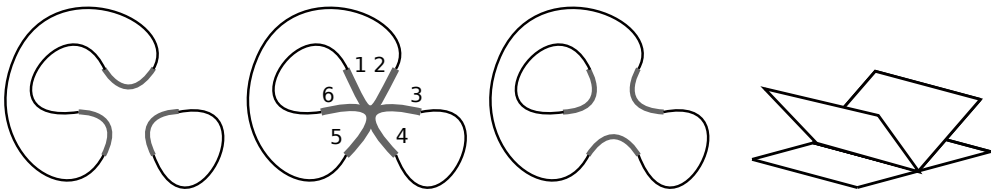
Note that this dihedral symmetry is only exhibited in the case of edge neighbourhoods. This is because the exit points of edges can be arranged consecutively in the edge star and because of how the inner fibers transition via the Exit Point Pair Swap Lemma (see Lemma 7.3).



(a) Computing the Reeb space of an indefinite edge of type three with global pairing $\{(1,2),(3,4),(5,6)\}$



(b) Computing the Reeb space of an indefinite edge of type three with global pairing $\{(1,4),(2,3),(5,6)\}$



(c) Computing the Reeb space of an indefinite edge of type three with global pairing $\{(1,2),(3,4),(5,6)\}$

Figure 8.5: An example of three equivalent global pairings. Each row shows the global fibers above, at and below an indefinite edge of type three. The first global fiber pairing is $\{(1,2),(3,4),(5,6)\}$ where those are the ids of the exit points. The second row is a rotation of that case, which yields a different global pairing, but an equivalent case. The third row is obtained by two rotations and is again equivalent to the first one.

8.2.3 Generating Edge Neighborhoods

For edges we have completely described the possible inner arrangements via the Exit Point Pair Swap Lemma (see Lemma 7.3) so we only need to generate all possible global arrangements. Our algorithm first generates all possible global arrangements, then reduces them by symmetry. For each global arrangement we compute the number of connected components in each region of the walk using the Exit Point Pair Swap Lemma 7.3. Finally we group all cases with equivalent Reeb spaces neighbourhoods. We recognize whether two cases have equivalent Reeb space neighbourhoods based on whether they have the same number of connected components in each region. This identifies the Reeb space neighbourhood of an edge uniquely because all fiber components merge in one singular components at the edge.

8.2.4 Generating Vertex Neighborhoods

For vertices we first need to generate all domain arrangements and all range arrangements. The domain arrangement consists of a triangulation of a sphere. For generating all topologically different triangulations of the sphere see Bowen & Fisk (1967).

Since all range arrangements can be described by a dihedral top permutation (DTP) all we have do is generate those. In order to generate all possible DTPs we generate all permutations on $n-1$ vertices. Since we know we can always start with v_0 we only need to permute the rest of the vertices. Then we generate all combinations and pick $1, 2, \ldots, n-2$ elements from the permutation to have bars or not. We exclude the element v_1 (as well as v_0) because we always choose it to not have a bar to eliminate mirror symmetries. In total for n vertices we have $(n-1)! \times 2^{(n-2)}$ distinct DTPs.

8.2.5 Reeb Space Data Structure

In Figure 8.4 we gave two possible representations of a Reeb space neighbourhood. The first one is the collection of sheets along with the information of how they attach to one another, the second is the Reeb graph of the walk. The issue with both of those data structures is that it's difficult to use them to compare two Reeb space neighbourhoods for equivalence. We need to add some additional information and segment them into regions.

We propose an alternative representation for local Reeb space neighbourhoods called a Reeb space signature graph, which we've demonstrated in Figure 8.6. Every node represents an equivalence class of fiber components either in regular or Jacobi regions (see Subsection 7.3.3). Black nodes stand for classes of regular fibers and white nodes for classes of singular fibers. Note that in Jacobi regions not all fiber components are singular, only one of them. The edges between nodes indicate how fiber components meet at a singular fiber, or how a component passes through a Jacobi edge without becoming singular.

Note that in some cases the regular Jacobi regions can be omitted to speed up computation. This is for example when there is no change in the fiber components with

a $1-1$ Jacobi edge (see Figure 9.3 fourth column) or when the change is unambiguous like in a $1-2$ or $1-3$ Jacobi edge (see Figure 9.3 first and second column). However in a case like $2-2$ (see Figure 9.3 third), we would lose information about how the sheets attach together if we omit the Jacobi region.

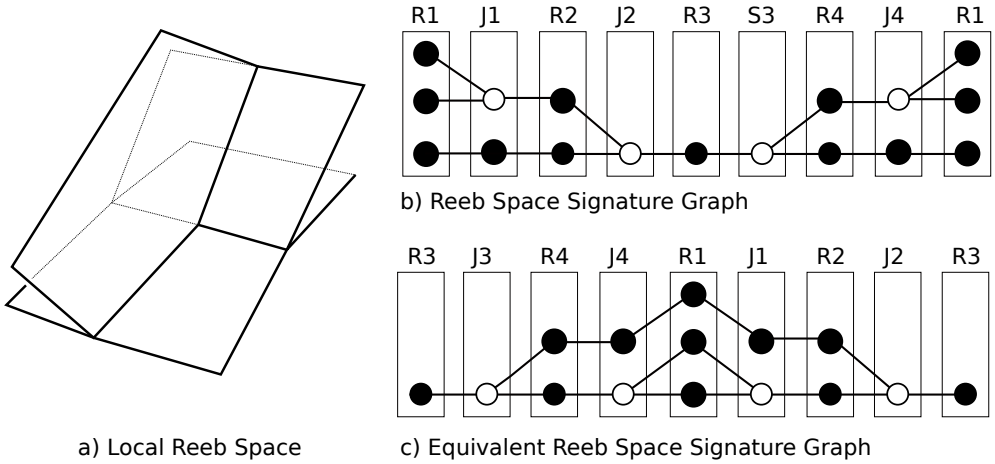


Figure 8.6: The data structure we use to represent Reeb space neighbourhoods is called a Reeb space signature graph. Black vertices represent regular classes of fibers and the white vertices represent singular classes of fibers. The vertices are grouped into regular and Jacobi regions. The starting region has been repeated at the end to allow a linear visualisation. Regular and Jacobi regions alternate as we move around in a walk around the center point. Regular regions are annotated as $R1, R2, R3, R4$ and Jacobi regions as $J1, J2, J3, J4$. Equivalent Reeb space neighbourhoods can be obtained via rotation or mirror of the signature graph or by permuting the vertices within each region.

Computing the Reeb Space Signature Graphs

In order to compare two Reeb space signature graphs we can can define an equivalence based symmetry and relabeling (see Figure 8.6 b)). Clearly any rotation or mirroring of the regions of a Reeb space signature graph gives us an equivalent one. Only their relative order matters. Additionally all the vertices within their respective regions can be permuted to obtain another equivalent signature graph. In order to compare two Reeb space signature graph we use the brute force technique of trying all possible rotations and mirrors as well as all possible permutations within each of the regions.

In order to compute the inner fiber arrangement we compute the combinatorial fibers in every region of the walk. We compute the combinatorial fibers by computing the active triangles using the regular-like DTP expressions (see Subsection 7.3.4). When we match a regular-like DTP expression we also consider its rotational and mirror symmetries. We match expressions to DTPs with a linear scan along the DTP. We find

the positions of each of the required vertices, determine if they are in the correct order and then determine whether they have a bar or not. This is done in time linear in the size of the DTP, but since we only consider meshes with bounded vertex degree that complexity is constant.

We only need to compute the combinatorial fibers of the regular regions. We can infer the connectivity of the fibers in the Jacobi regions based on which exit points pairs have changed. The ones that have changed have all their fiber components merged into a single component in the Jacobi region. For definite edges we consider the fiber born in the adjacent regular region, not in the Jacobi region where it is a point.

Once we have the combinatorial fibers in each region we build up the Reeb space signature graph by considering all adjacent regions. Using the union find data structure we can determine which fiber components have merged/split with which other components. This gives us the edges between the regular and Jacobi regions in the Reeb space signature graph.

Overall the computation of each Reeb space signature graph is linear in the number of regions, which is bounded by the vertex degree. The computational bottleneck is considering all possible cases and reducing them. That means considering all combinations of domain, range and global fiber arrangements and then reducing all Reeb space signature graphs to a unique set with a single representative using all possible mirror and relabeling symmetries.

Chapter 9

Results

9.1 Reeb Space Application Suite

In Section 8.2 we discussed methods to generate all possible Reeb space local neighbourhoods. Those methods involved generating all possible domain and range arrangements to obtain the local arrangement, then combining that with the outer fiber arrangement to obtain the global fiber arrangement and thus the local Reeb space neighbourhood. In this section we will discuss the applications we've developed to implement those methods.

We have developed two key applications in the course of this research. The first one is for visualising the local neighbourhoods of vertices and edges in the range and in the domain. The application allows us to select fiber points and visualise their fiber components. This application was useful in obtaining intuition when developing the theory presented in this part as well as manually debugging the algorithms for generating local neighbourhoods. The second application generates all possible local Reeb space neighbourhoods by generating and combining all the domain and range arrangements as well as the inner and outer fiber arrangements.

9.1.1 Fiber Visualiser

The goal of the fiber visualiser application is to visualise the geometry of fibers. You can see an example of this application in Figure 9.1. The left hand view is of the simplicial complex in the domain as well as the selected fiber components in red. The fiber is computed geometrically per tetrahedron as discussed in Section 7.1. The left hand view can be rotated as well as zoomed in/out. The right hand view is the image of the simplicial complex in the range, where the images of all vertices are labeled by their ID. The controls at the bottom have to do with opacity of the vertices, edges, triangles and fibers in the domain.

As input, the Fiber Visualiser takes a Reeb space data file, of which we've shown an example in Figure 9.2. The Reeb space data file consists of the domain 3D coordinates

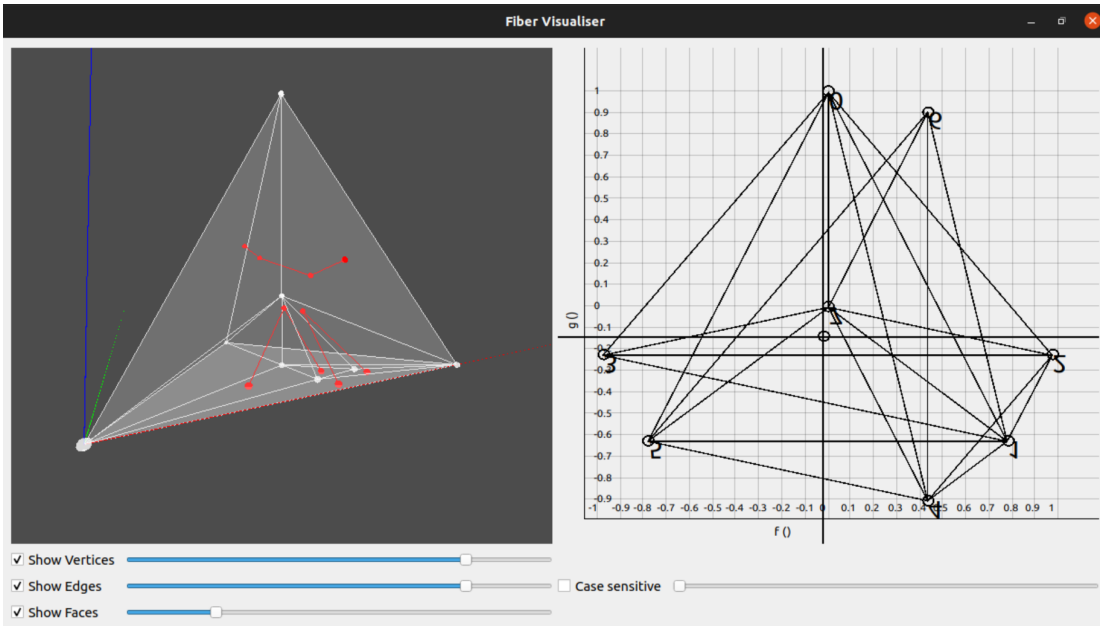


Figure 9.1: The fiber visualiser application as it is used to visualiser fibers at a particular fiber point. Left: a view of the geometry in the domain and the fiber (in red). Right: a view of the geometry in the range and the fiber point (in the crosshair). Bottom: controls for setting opacity and other utilities.

of the vertices, the 2D coordinates of where they map to in the range, as well as a list of all tetrahedra with their vertex IDs. In the examples we use we set the domain coordinates and list of tetrahedra manually. In the case of vertex neighbourhoods we generate the vertex range coordinates by setting them on the unit circle around the center vertex using a particular dihedral top permutation. For the case of edges, we also set them on a circle around the diameter ab and we also use a DTP to set which vertices are above and which are below the edge.

The fiber visualiser application has been an integral part of this research for multiple reasons. The first one is that it allows us to obtain a basic visual understanding behind how fibers of PL functions behave. Using the application we can identify active triangles and construct combinatorial and geometric fibers. This allows us to manually construct local Reeb space neighbourhoods for any domain and range configuration. This has been important in developing our intuition and in debugging and testing our application for automatically generating Reeb space signature graphs.

A tool like the Fiber Visualiser can also be useful for a guided introduction to the theory of PL Reeb space data analysis. The application is written in C++ using QT 5.4 and OpenGL.

```
# The first two integers are the number of vertices and number of tetrahedra.
8 6

# Next we have the domain positions of the vertices in R3.
0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1

# Next we have the range positions of the vertices in R2.
9 8
1 4
4 9
3 3
7 9
0 2
6.5 3.5
0 8

# Finally we have a list of all the tetrahedra.
0 7 1 3
0 7 1 5
0 7 5 4
0 7 2 3
0 7 2 6
0 7 4 6
```

Figure 9.2: This is an example of a Reeb space data file. In includes information about the coordinates of the vertices in the domain and in the range as well as the connectivity of the mesh in the domain.

9.1.2 Generate Local Reeb Space Neighborhoods

The second application generates all unique local Reeb space neighbourhoods for a given domain and range arrangement using the methods we discussed in Section 8.2. The first stage in the computation is to use the domain and range arrangement to generate the inner fiber pairings of the walk. The next step is to generate all the outer exit point pairings based on the number of exit points. Then we compute the local Reeb space signature graphs for each outer point pairing. Finally we reduce the local Reeb space signature graphs based on rotational, mirror and permutation symmetry.

9.2 Results

Finally we present a classification of Reeb space neighbourhoods for low degree meshes as a proof of concept to illustrate how our method differs from previous ones. This section discusses our findings for meshes with maximum edge and vertex degree six. By vertex degree we mean the number of adjacent edges and by edge degree we mean

number of adjacent tetrahedra. We use the algorithms we described in Chapter 8.2 and implemented in Section 9.1 to generate all possible Reeb space neighbourhoods and then reduce them by mirror and relabeling symmetry (see Subsection 8.2.5) to a unique set.

First we will discuss our findings for the Reeb space neighbourhood of a single edge. We categorise the cases into those that can be realised in orientable meshes and those that cannot - orientable and non-orientable. Then we describe the Reeb space neighbourhoods of two intersecting edges and finally for vertices. We categorise those as orientable or not based on the types of edges they contain.

9.2.1 One Edge Neighborhoods

We first show the results for the Reeb space neighbourhoods of a single edge in Figure 9.3. The table shows the transition of the fibers as we cross a Jacobi edge as well as the resulting Reeb space. Recall that the fiber above the edge transitions to the singular fiber at the edge and then to the regular fiber below the edge. At the bottom of the table we have shown the Reeb space neighbourhood. Only the first column of the orientable cases has been previously described in the PL literature Edelsbrunner *et al.* (2008c).

In Figure 9.3 not all Reeb spaces are unique. For example there are two ways we can obtain the Reeb space where one sheet splits off into two sheets (the first and last column). However in the first column this happens for an orientable mesh and an indefinite edge of type two and in the last column this cannot happen in an orientable mesh for an indefinite edge of type three. We see that knowing about the orientability and the edge degree of edges in the mesh tell us about the realisable Reeb space neighbourhoods. Considering only orientable meshes reduces reduces the number of cases we need to consider.

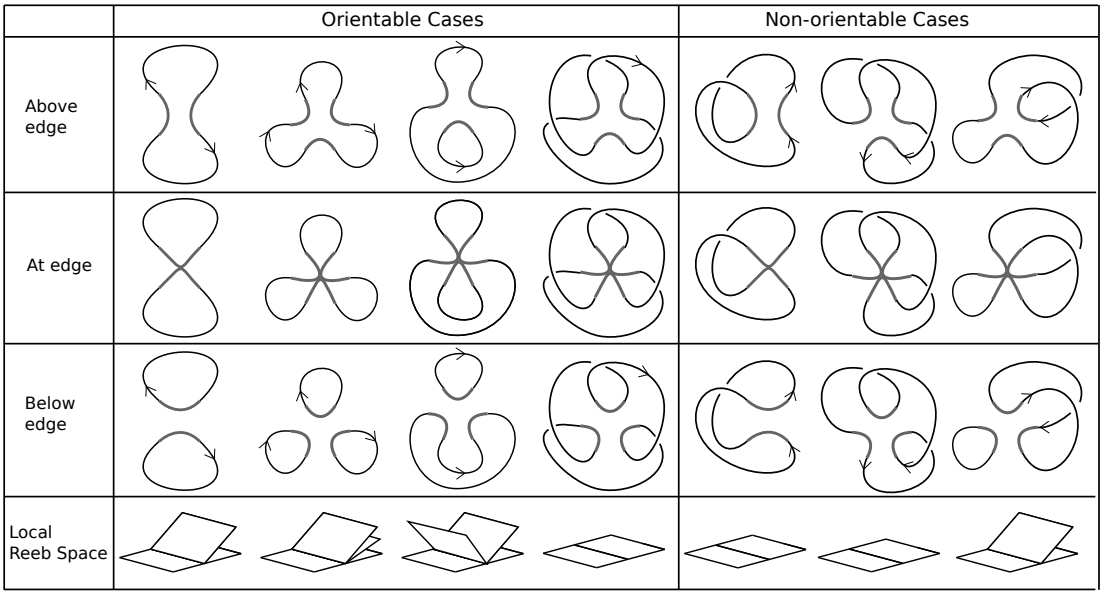


Figure 9.3: All possible Reeb space neighbourhoods for a single edge of degree up to six (up to six adjacent tetrahedra in the mesh). We compute the Reeb space by taking three fibers - one at the Jacobi edge (at edge) and two on either side of the Jacobi edge (above and below edge). The number of sheets in the Reeb space corresponds to the number of connected components in each of the three fibers. We have split the classification into orientable - those that can only be realised in orientable meshes, and non-orientable - those that can be realised in non-orientable meshes as well. To recognize the orientable cases we place any orientation on the regular fiber in the above case and see if the orientation remains consistent after passing through the singular fiber into the below case Levine (1988). Follow the arrows on the fiber (the orientation) for non-orientable fibers to see that it is is no longer consistent. The cases where more than two fibers meet at a Jacobi edge as well as the non-orientable cases have not been described in the literature previously.

Another observation we can make is about the cases where a single sheet remains a single sheet as we cross the edge. For indefinite edges of type two this cannot happen in orientable meshes, but for indefinite edges of type three this can happen in both orientable and non-orientable meshes. This highlights the value of grouping the cases by dihedral symmetry, otherwise we would not have discovered this behaviour.

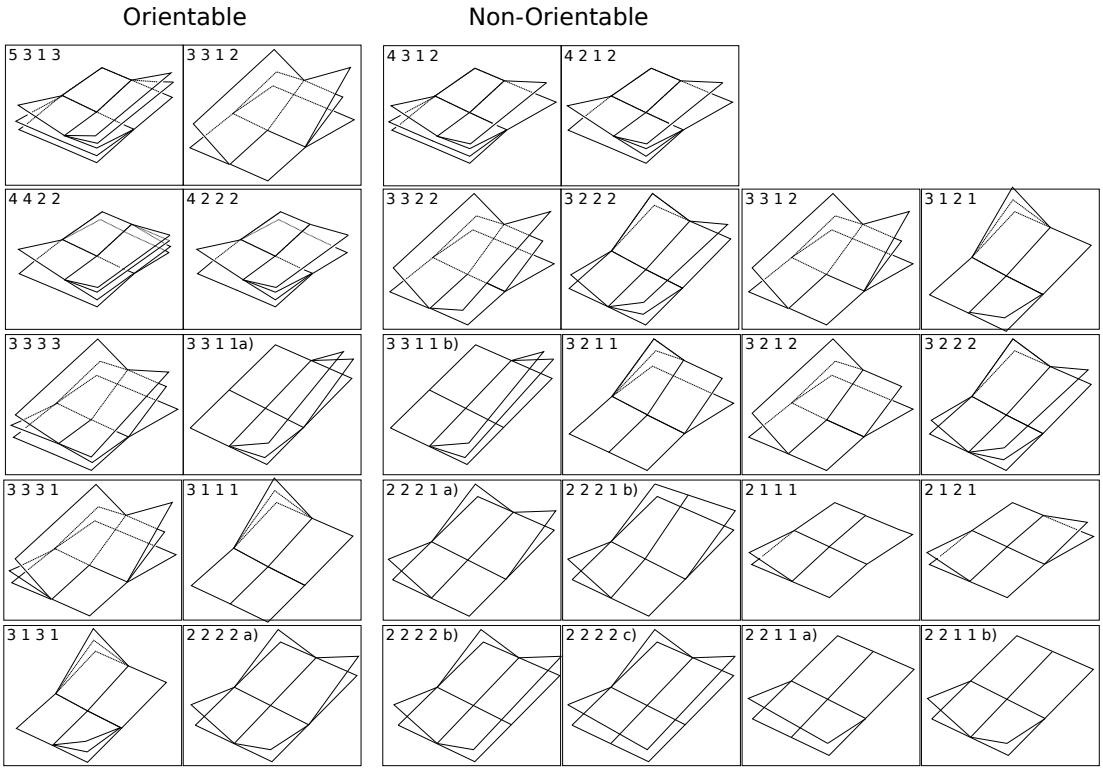


Figure 9.4: The possible Reeb space neighbourhoods for when the images of two indefinite Jacobi edges of type three intersect in the range. In this work we extend the indefinite edge type to indefinite edge of type n where $n \in \{2, 3, \ldots, \infty\}$. An indefinite Jacobi edge of type n is when n fiber component merge/split apart at that edge (much like saddles in the scalar case). Previous work which had identified two cases for edge intersection for orientable cases under strict conditions on the PL mapping. When we relax the condition of the PL map to be more practical for data analysis we get a much more varied number of cases. The ID of each case lists the number of fiber components in each quadrant.

9.2.2 Two Edges Neighborhoods

When the images of two indefinite edges intersect in the range there are several possible cases. It could be either two indefinite edges of type two, two indefinite edges of type three or one of each type. In Figure 9.4 we have shown all the possible configurations for when the images of two indefinite edges of type three intersect.

We note that the Reeb space neighbourhoods in Figure 9.4 also include the other two cases of indefinite edges of type two and three intersecting. The only difference is whether some cases are orientable or not. This is because when one of the edges is indefinite edge of type two we can have the $2 - 1$ transition of fiber components.

The $2-1$ case (see Figure 9.3) is orientable for indefinite edges of type two, but non-orientable for indefinite edges of type three. Likewise for the edges of type $1-1$ as we discussed in the previous subsection.

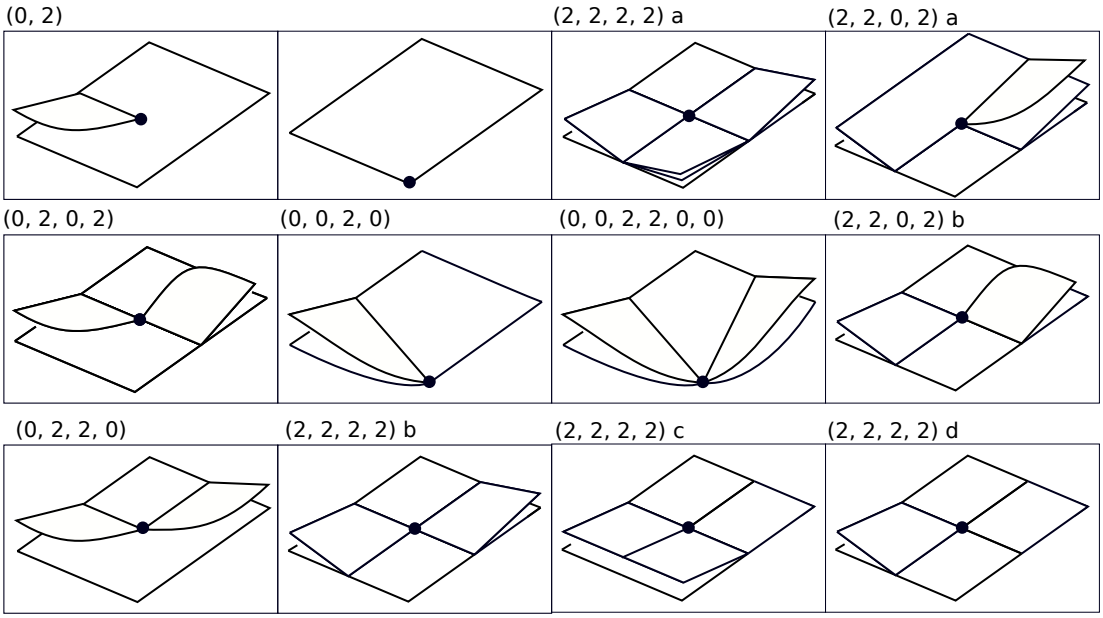


Figure 9.5: All realisable Reeb space neighbourhoods for a generic piecewise linear mapping (See Definition 6.1) over a closed piecewise linear 3-manifold with maximum vertex degree six. All cases are labeled with the Jacobi edge type of all incident edges in a clockwise order. In this notation 0 stands for a definite edge (extrema, a curve appears or disappears) and 2 for an indefinite edge of type two (saddle, two fiber components meet at that edge). Only the first two cases in the first row have been described for vertex neighbourhoods in the PL case in previous literature. Note that the next two cases $(2,2,2,2)$ a) and $(2,2,2,2)$ b) have been described, but only for the neighbourhoods of the intersection of two edges in the range, not for vertex neighbourhoods. All cases are orientable besides $(2,2,2,2)$ c) and $(2,2,2,2)$ d) because they include a $1-1$ transition which is not realisable in orientable meshes for indefinite edges of type two.

9.2.3 Vertex Neighborhoods

The results for Reeb space vertex neighbourhoods are shown in Figure 9.5. We note that while there are two topologically different triangulation of the sphere on six vertices, but that both generate the same Reeb space neighbourhoods. In the top row we have shown the cases that have appeared previously in the PL literature Edelsbrunner *et al.* (2008c).

The IDs of the cases list the Jacobi types of the edges adjacent to the vertex. We use 0 for definite edges and 2 for indefinite edges of type two. Previous we identified cases by the number of fiber components, but it's more indicative to list the Jacobi edge types for vertices. This is because various types of Jacobi edges can meet at a vertex and that determines the Jacobi type of the vertex.

Traditionally in the literature Saeki (2017) the case where an indefinite and definite edge meet is called a cusp. In our notation this would be the $(0, 2)$ case in Figure 9.5. However with our method we can have many combinations of edge types meeting at a vertex and we obtain case such as $(0, 2, 0, 2)$ or $(0, 0, 2, 0)$. Hence with our definition of generic the term cusp isn't as indicative and it isn't clear whether it should be used or generalised.

We note that that cases $(2, 2, 2, 2)$ a)-d) also appear as Reeb space neighbourhoods for the intersections two edges. Intuitively this makes sense since $(2, 2, 2, 2)$ is like an intersection of two indefinite edges of type two. However with the vertex case those edges are all adjacent and the vertex star has only six exit points. By contrast the edge case has eight exit points and the two edges are not adjacent in the mesh.

Part V

Conclusions and Future Work

Chapter 10

Conclusions

This thesis has made three contributions to the field of topological data analysis for scientific visualisation. These contributions range from algorithmic, in improving the scalability of algorithms for computing scalar topological data structures; to applied, in developing tools and techniques to visualise bivariate and scalar data; to theoretical, in developing the mathematics behind bivariate topological data structures.

10.1 Hypersweeps

The first contribution of this thesis is the implementation scalable techniques to utilise the contour tree in data-parallel on a single computer. These techniques include computing geometric measures, simplification, branch decomposition and single contour extraction. Previous work had only focused on parallelising contour tree computation, but not these secondary operations. This limited the applicability of the contour tree for real world data analysis.

To enable our scalable solution, we introduced the *hypersweep* - a data-parallel method for computing properties in contour trees, based on parallel tree contraction. We implemented the hypersweep in the open source VTK-m library and used it to develop an *in situ* visualisation pipeline using the Cinema database. We achieved a speedup on up to 7x against serial code on data sets of up to one billion elements.

10.2 Convective Clouds

The second major contribution is the development of a visualisation application to assist atmospheric scientists in understanding convective cloud triggering in the Earth's atmosphere. The novelty of this application is in updating link and brush techniques by combining scalar and bivariate visualisation tools. We decompose features based on the intersection of the volume contained in a fiber surface and the super-level set of an isosurface with a novel technique we called Cartesian Fiber Surfaces.

The tool we developed has been used extensively by a domain scientist from the School of Earth of Environment at the University of Leeds and it is now a standard part of their workflow. Up until then the workflow of the scientist involved analysing their features of interest using multiple two-dimensional non-interactive plots. Our tool not only speeds up the scientist's workflow but it also allowed them for the first time to see their features authentically, in three dimensiosn. In addition to this, the use of the tool has lead to findings about areas of the environment which encourage and inhibit cloud formation and about the internal structure of cloud triggering features.

10.3 Reeb Spaces

The third and final major contribution is the introduction of algorithms for generating and classifying the local structure of the higher dimensional generalisation of the contour tree called the Reeb space. Describing that structure is important in developing more efficient algorithms for computing and utilising the Reeb space. Previous methods either do not describe the possible Reeb space structure explicitly or describe it for a limited type input data. Our method relaxes the assumptions about the type of input data to be more in line with practical data analysis considerations.

To describe the local structure of Reeb spaces we developed combinatorial algorithms that exhaust all possible configurations, while considering geometric symmetry. In addition to the previously considered in the literature Jacobi edges we introduced Jacobi vertices as meeting points of multiple Jacobi edges. We showed that we can generate all types of Jacobi edges and vertices and presented a classification algorithm for meshes with bounded edge and vertex degree. Finally, we presented a full classification for meshes with maximum vertex and edge degree six.

Chapter 11

Future Work

In this chapter we will describe over ideas for future work for the three topics of this thesis. We will go over some of the main limitations and how they can be addressed.

11.1 Hypersweeps

The main limitation of the work on parallelising the secondary contour tree operations is that it only scales on a single computer. This becomes an issue because many data sets nowadays are approaching petascale and exascale sizes, where they cannot be stored on a single node in a computing cluster. Since the hypersweep is designed with shared memory parallelism in mind it would be difficult to apply to those data sets directly. One major direction of future work will be to work a distributed version of the hypersweep based on the already existing distributed contour tree computation Carr *et al.* (2021a).

The second major research direction would be to adapt the hypersweep to compute the full suite of geometric and statistical measures as described in Carr *et al.* (2010c). Those include volume, persistence, surfaces, integrated function value (hypervolume), mean value and variance. Furthermore, if it is not feasible to compute persistence as originally described in Pascucci *et al.* (2004a) and Carr *et al.* (2010c), then we would like to perform an extensive evaluation on our proposed height measure. Identifying the differences between the two and outlining general guidelines about which metric to use is vital for practical contour tree data analysis.

11.2 Convective Clouds

In our work with atmosphere scientists on understanding structures in the atmosphere that trigger convective cloud formation we could not describe the scientist's workflow in enough detail. This is important in evaluating the software tool we have developed and determining what next steps to take in improving it. We aim to describe the scientists'

workflow before and after introducing the tool and documenting how and why the tool improves that.

Additionally, using the tool we only studied a single cloud triggering feature in isolation. The next step in this line or work is to extend the software tool we have developed to study multiple features over time. This would be greatly aided by tools that compare topological structures of the scalar and the bivariate field. Those tools rely on the further development of higher dimensional topological data structures such as the Reeb Space in order to identify and track features defined in more than one scalar field.

11.3 Reeb Space Neighbourhoods

A practical extension to our work on describing Reeb space neighbourhoods would be to describe those for meshes with a boundary. First, this would involve describing fiber behaviour of the neighbourhoods of vertices and edges in the input mesh where those vertices and edges are part of the boundary of the mesh. This can be done with the same method we have described, as long as we enumerate all types of boundary edge and vertex neighbourhoods. Secondly, this would involve extending local fibers to global fibers by pairing exit points with boundary points. An example of such global pair would be $\{(1, B), (2, 3), (4, 5), (6, B)\}$ where B means that the fiber intersects the boundary.

Another extension would be to describe the Reeb space neighbourhoods of meshes which are not tetrahedral. This work was outside of the scope of this thesis because the fiber behaviour in meshes with hexahedral cells Klacansky *et al.* (2017b) becomes increasingly complex. Under a trilinear interpolant fibers are the intersection of two isosurfaces which are hyperbolic sheets. This further increases the complexity of our approach because we also have to combinatorially describe the behaviour of fibers withing each individual cell.

The major limitation of our approach is that it relies on the combinatorial enumeration of all possible neighbourhoods in the range and in the domain to generate all possible Reeb spaces. As we have seen the number of possible neighbourhoods is much larger than the resulting number of possible Reeb spaces. We would like to investigate a way of deriving the possible Reeb spaces directly in a more efficient way. This would involve describing a pattern that relates the possible Reeb spaces to the mesh vertex and edge degree. A first step in doing so would be go generate Reeb space neighbourhoods of higher degree meshes, work out some relations by hand and then attempt to prove a generalisation of those.

Furthermore, we would like to investigate the question of how our combinatorial description of Reeb space neighbourhoods can be used towards to Reeb space computation algorithm. The first step in contour tree computation is to apply simulation of simplicity to differentiate vertices with equal function value by giving them a unique integer index. The algorithm then proceeds using the index of each vertex and is entirely combinatorial. For a bivariate case we cannot directly sort all vertices by function

value, but we can use the dihedral top permutation of each vertex to disambiguate the order of all vertices. Our next step would be determine how to extend the local arrangement of vertices and how their DTPs can be extended to a global arrangement to enable more efficient combinatorial algorithms.

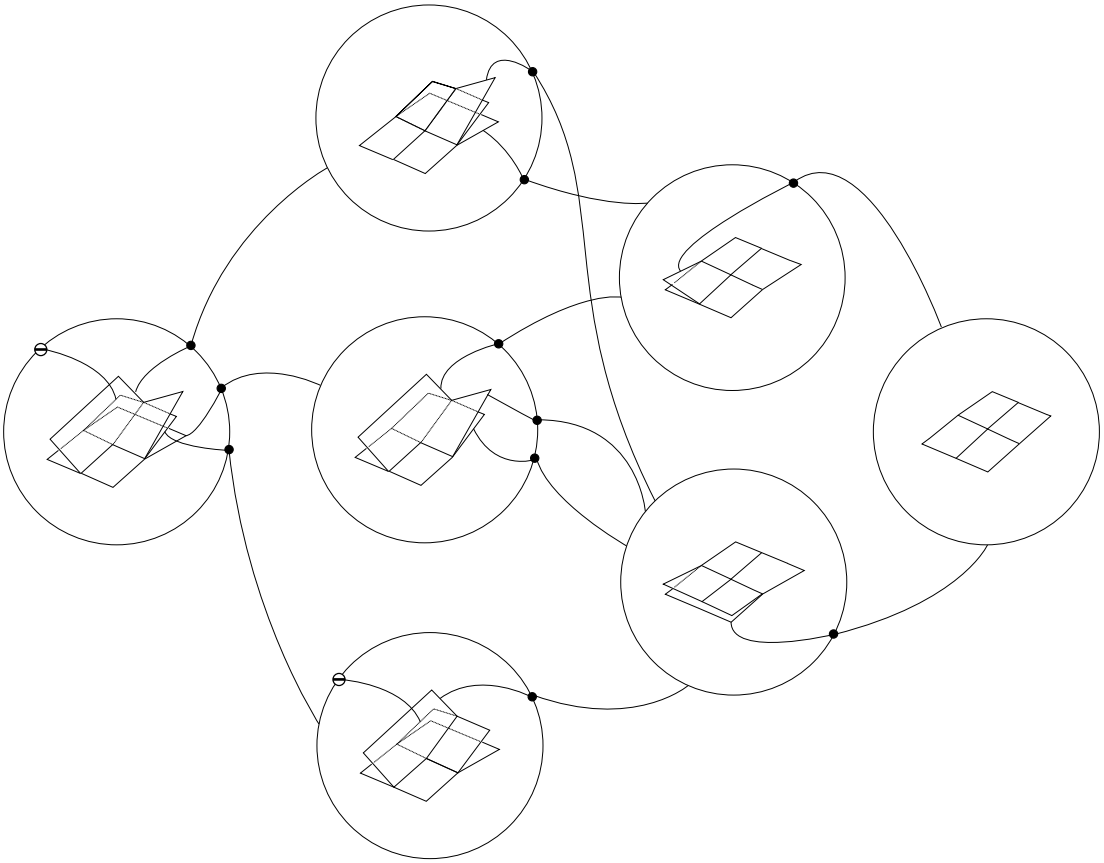


Figure 11.1: A diagram of how we can use the enumeration of Reeb graph neighbourhood to perform local sheet simplification. When a sheet is remove the diagram shows us which other local case we transition to. When there is no such case, removing that is not permitted.

Finally, our work also lays the foundation for developing combinatorial algorithms for Reeb space and simplification. In fact the tables we presented in the results section (see Chapter 9.2) can be used for reference for Reeb Space simplification. We can only simplify (or remove a sheet) from a local Reeb space neighbourhood if that sheet leaves us in another case that is in the table. Otherwise that simplification operation should be prohibited, because such a Reeb space neighbourhood is not realisable. We give an example of how such local simplification can be applied in Figure 11.1.

References

(2020). The Topology TooKit. https://topology-tool-kit.github.io/, accessed: 2020-12-04. 75

Acharya, A. & Natarajan, V. (2015). A parallel and memory efficient algorithm for constructing the contour tree. In *2015 IEEE Pacific Visualization Symposium (PacificVis)*, 271–278. 24

Agarwal, P.K., Edelsbrunner, H., Harer, J. & Wang, Y. (2006). Extreme elevation on a 2-manifold. *Discrete & Computational Geometry*, **36**, 553–572. 25

Ahrens, J., Jourdain, S., O'Leary, P., Patchett, J., Rogers, D.H. & Petersen, M. (2014). An Image-Based Approach to Extreme Scale in Situ Visualization and Analysis. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 424–434. 46

Artamonova, I.V., Alekseev, V.V. & Makarenko, N.G. (2017). Gradient measure and jacobi sets for estimation of interrelationship between geophysical multifields. *Journal of Physics: Conference Series*, **798**, 012040. 31

Attali, D., Glisse, M., Hornus, S., Lazarus, F. & Morozov, D. (2009). Persistence-sensitive simplification of functions on surfaces in linear time. *Presented at TOPOINVIS*, **9**, 23–24. 25

Attene, M., Biasotti, S., Mortara, M., Patanè, G., Spagnuolo, M. & Falcidieno, B. (2006). Computational methods for understanding 3d shapes. *Computers & Graphics*, **30**, 323 – 333. 21

Ayachit, U. (2015). *The ParaView Guide: A Parallel Visualization Application*. Kitware, Inc., USA. 39

Bachthaler, S. & Weiskopf, D. (2008a). Continuous scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, **14**, 1428–1435. 29

Bachthaler, S. & Weiskopf, D. (2008b). Continuous Scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, **14**, 1428–1435. 76

Bajaj, C.L., Pascucci, V. & Schikore, D.R. (1997). The contour spectrum. In *Proceedings. Visualization '97 (Cat. No. 97CB36155)*, 167–173. 21

Basu, S., Cox, N. & Percival, S. (2018). On the reeb spaces of definable maps. *ArXiv e-prints*. 31

Bauer, U., Lange, C. & Wardetzky, M. (2012). Optimal topological simplification of discrete functions on surfaces. *Discrete & Computational Geometry*, **47**, 347–377. 25

Bennett, J., Pascucci, V. & Joy, K. (2007). Genus oblivious cross parameterization: Robust topological management of inter-surface maps. In *15th Pacific Conference on Computer Graphics and Applications (PG'07)*, 238–247. 31

Bhatia, H., Wang, B., Norgard, G., Pascucci, V. & Bremer, P.T. (2015). Local, smooth, and consistent jacobi set simplification. *Computational Geometry*, **48**, 311 – 332. 30, 31, 36

Biasotti, S., Falcidieno, B. & Spagnuolo, M. (2000). Extended reeb graphs for surface understanding and description. In G. Borgefors, I. Nyström & G.S. di Baja, eds., *Discrete Geometry for Computer Imagery*, 185–197, Springer Berlin Heidelberg, Berlin, Heidelberg. 21

Biedert, T. & Garth, C. (2015). Contour Tree Depth Images For Large Data Visualization. In C. Dachsbacher & P. Navrátil, eds., *Eurographics Symposium on Parallel Graphics and Visualization*, The Eurographics Association. 53

Bowen, R. & Fisk, S. (1967). Generation of triangulations of the sphere. *Mathematics of Computation*, **21**, 250–252. 116

Bremer, P.., Hamann, B., Edelsbrunner, H. & Pascucci, V. (2004). A topological hierarchy for functions on triangulated surfaces. *IEEE Transactions on Visualization and Computer Graphics*, **10**, 385–396. 25

Bremer, P., Bringa, E., Duchaineau, M., Gyulassy, A., Laney, D., Mascarenhas, A. & Pascucci, V. (2007). Topological feature extraction and tracking. In *Journal of Physics: Conference Series*, vol. 78, 012007, IOP Publishing. 31, 36, 38

Bremer, P., Weber, G., Tierny, J., Pascucci, V., Day, M. & Bell, J. (2011). Interactive exploration and analysis of large-scale simulations using topology-based data segmentation. *IEEE Transactions on Visualization and Computer Graphics*, **17**, 1307–1324. 37

Burlet, O. (1974). Sur certaines applications génériques d'une variété close à 3 dimensions dans le plan. *Enseign. Math.*, **20**, 275–292. 31

Carlsson, G. & Zomorodian, A. (2009). The theory of multidimensional persistence. *Discrete & Computational Geometry*, **42**, 71–93. 21

Carlsson, G., Danciger, J. & Morton, J. (2009a). Mapping geometry in heart rate data. 34

Carlsson, G., Singh, G. & Zomorodian, A. (2009b). Computing multidimensional persistence. In Y. Dong, D.Z. Du & O. Ibarra, eds., *Algorithms and Computation*, 730–739, Springer Berlin Heidelberg, Berlin, Heidelberg. 21

Carr, H. & Duke, D. (2014). Joint contour nets. *IEEE Transactions on Visualization and Computer Graphics*, **20**, 1100–1113. 34

Carr, H., Snoeyink, J. & Axen, U. (2000). Computing contour trees in all dimensions. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 918–926. 22

Carr, H., Moller, T. & Snoeyink, J. (2001). Simplicial subdivisions and sampling artifacts. In *Proceedings Visualization, 2001. VIS '01.*, 99–547. 19

Carr, H., Snoeyink, J. & Axen, U. (2003). Computing Contour Trees in All Dimensions. *Computational Geometry: Theory and Applications*, **24**, 75–94. 23, 24, 75

Carr, H., Snoeyink, J. & van de Panne, M. (2010a). Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree. *Computational Geometry*, **43**, 42 – 58, special Issue on the 14th Annual Fall Workshop. 25, 37

Carr, H., Snoeyink, J. & van de Panne, M. (2010b). Flexible Isosurfaces: Simplifying and Displaying Scalar Topology Using the Contour Tree. *Computational Geometry: Theory and Applications*, **43**, 42–58. 43, 44, 46, 50

Carr, H., Snoeyink, J. & Van De Panne, M. (2010c). Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree. *Computational Geometry*, **43**, 42–58. 130

Carr, H., Geng, Z., Tierny, J., Chattopadhyay, A. & Knoll, A. (2015). Fiber surfaces: Generalizing isosurfaces to bivariate data. *Computer Graphics Forum*, **34**, 241–250. 27, 29

Carr, H., Weber, G.H., Sewell, C. & Ahrens, J. (2016). Parallel Peak Pruning for Scalable SMP Contour Tree Computation. In *6th IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, 75–84. 45

Carr, H., Weber, G.H., Sewell, C., Rübel, O., Fasel, P. & Ahrens, J. (2019). Scalable Contour Tree Computation by Data Parallel Peak Pruning. *IEEE Transactions on Visualization and Computer Graphics*, 1–1. 11, 41, 45, 52, 54, 57

Carr, H., Rübel, O. & Weber, G.H. (2021a). Distributed Hierarchical Contour Trees. In *12thth IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. 130

Carr, H., Rubel, O., Weber, G.H. & Ahrens, J. (2021b). Optimization and augmentation for data parallel contour trees. *IEEE transactions on visualization and computer graphics*. 41, 42, 48, 52, 55, 57

Carr, H.A., Weber, G.H., Sewell, C.M. & Ahrens, J.P. (2016). Parallel peak pruning for scalable SMP contour tree computation. In *2016 IEEE 6th Symposium on Large Data Analysis and Visualization (LDAV)*, 75–84. 23, 24

Carrière, M. & Oudot, S. (2017). Structure and stability of the one-dimensional mapper. *Foundations of Computational Mathematics*. 33

Carrière, M., Michel, B. & Oudot, S. (2018). Statistical analysis and parameter selection for mapper. *Journal of Machine Learning Research*, **19**, 1–39. 33

Cazals, F., Chazal, F. & Lewiner, T. (2003). Molecular shape analysis based upon the morse-smale complex and the connolly function. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, SCG '03, 351–360, ACM, New York, NY, USA. 20

Cerri, A. & Landi, C. (2013). The persistence space in multidimensional persistent homology. In R. Gonzalez-Diaz, M.J. Jimenez & B. Medrano, eds., *Discrete Geometry for Computer Imagery*, 180–191, Springer Berlin Heidelberg, Berlin, Heidelberg. 21

Chattopadhyay, A., Carr, H., Duke, D. & Geng, Z. (2014). Extracting jacobi structures in reeb spaces. *Eurographics Conference on Visualization (EuroVis)*. 32

Chattopadhyay, A., Carr, H., Duke, D., Geng, Z. & Saeki, O. (2016). Multivariate topology simplification. *Computational Geometry*, **58**, 1 – 24. 37

Cohen-Steiner, D., Edelsbrunner, H. & Morozov, D. (2006). Vines and vineyards by updating persistence in linear time. In *Proceedings of the twenty-second annual symposium on Computational geometry*, 119–126, ACM. 38

Cohen-Steiner, D., Edelsbrunner, H. & Harer, J. (2007). Stability of persistence diagrams. *Discrete & Computational Geometry*, **37**, 103–120. 33

Cole-McLaughlin, K., Edelsbrunner, H., Harer, J., Natarajan, V. & Pascucci, V. (2004). Loops in reeb graphs of 2-manifolds. *Discrete & Computational Geometry*, **32**, 231–244. 21

Coudriau, M., Lahmadi, A. & François, J. (2016). Topological analysis and visualisation of network monitoring data: Darknet case study. In *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, 1–6. 34

Couvreux, F., Hourdin, F. & Rio, C. (2010). Resolved Versus Parametrized Boundary-Layer Plumes. Part I: A Parametrization-Oriented Conditional Sampling in Large-Eddy Simulations. *Boundary-Layer Meteorology*, **134**. 63, 67

Danovaro, E., Floriani, L.D. & Vitali, M. (2007). Multi-resolution morse-smale complexes for terrain modeling. In *14th International Conference on Image Analysis and Processing (ICIAP 2007)*, 337–342. 20

De Cecco, L., Nicolau, M., Giannoccaro, M., Daidone, M.G., Bossi, P., Locati, L., Licitra, L. & Canevari, S. (2015). Head and neck cancer subtypes with biological and clinical relevance: Meta-analysis of gene-expression data. *Oncotarget*, **6**, 9627–9642. 34

Delfinado, C.J.A. & Edelsbrunner, H. (1995). An incremental algorithm for betti numbers of simplicial complexes on the 3-sphere. *Computer Aided Geometric Design*, **12**, 771 – 784, grid Generation, Finite Elements, and Geometric Design. 30

Denby, L., Böing, S., Parker, D. & Tobias, S. (2019). The effect of ambient shear on coherent boundary layer structures. *Journal of Atmospheric Sciences*, To appear. 67

Denby, L., Böing, S.J., Parker, D.J., Ross, A.N. & Tobias, S.M. (2022). Characterising the shape, size, and orientation of cloud-feeding coherent boundary-layer structures. *Quarterly Journal of the Royal Meteorological Society*, **148**, 499–519. 12, 66

Dey, T.K., Mémoli, F. & Wang, Y. (2015). Mutiscale mapper: A framework for topological summarization of data and maps. *CoRR*, **abs/1504.03763**. 33

Dey, T.K., Memoli, F. & Wang, Y. (2017). Topological analysis of nerves, reeb spaces, mappers, and multiscale mappers. *ArXiv e-prints*. 34

Doraiswamy, H. & Natarajan, V. (2009). Efficient algorithms for computing reeb graphs. *Computational Geometry*, **42**, 606–616. 22

Doraiswamy, H. & Natarajan, V. (2012). Output-sensitive construction of reeb graphs. *IEEE Transactions on Visualization and Computer Graphics*, **18**, 146–159. 22

Doraiswamy, H. & Natarajan, V. (2013). Computing reeb graphs as a union of contour trees. *IEEE Transactions on Visualization and Computer Graphics*, **19**, 249–262. 22

Doraiswamy, H., Natarajan, V. & Nanjundiah, R.S. (2013). An exploration framework to identify and track movement of cloud systems. *IEEE Transactions on Visualization and Computer Graphics*, **19**, 2896–2905. 39

Duke, D.J., Carr, H.A., Knoll, A., Schunck, N., Nam, H.A. & Staszczak, A. (2012). Visualizing nuclear scission through a multifield extension of topological analysis. *IEEE Trans. Vis. Comput. Graph.*, **18**, 2033–2040. 34

Edelsbrunner, Harer & Zomorodian (2003a). Hierarchical morse—smale complexes for piecewise linear 2-manifolds. *Discrete & Computational Geometry*, **30**, 87–107. 20

Edelsbrunner, H. & Harer, J. (2002a). Jacobi sets of multiple morse functions. *Foundations of Computational Mathematics, Minneapolis*, 37–57. 29, 30

Edelsbrunner, H. & Harer, J. (2002b). Jacobi sets of multiple Morse functions. *Foundations of Computational Mathematics, Minneapolis*, 37–57, publisher: Cambridge Univ. Press. 84, 85, 87, 90, 92

Edelsbrunner, H. & Harer, J. (2008). Persistent homology-a survey. *Contemporary mathematics*, **453**, 257–282. 19

Edelsbrunner, H. & Harer, J. (2010). *Computational topology: an introduction*. American Mathematical Soc. 87

Edelsbrunner, H. & Harer, J.L. (2022). *Computational topology: an introduction*. American Mathematical Society. 15

Edelsbrunner, H. & Mücke, E.P. (1990). Simulation of Simplicity: A Technique to Cope with Degenerate Cases in Geometric Algorithms. *ACM Transactions on Graphics*, **9**, 66–104. 87

Edelsbrunner, H. & Parsa, S. (2014). On the computational complexity of betti numbers: Reductions from matrix rank. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, 152–160, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. 20

Edelsbrunner, H., Letscher, D. & Zomorodian, A. (2000). Topological Persistence and Simplification. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, 454–463. 43

Edelsbrunner, H., Harer, J., Natarajan, V. & Pascucci, V. (2003b). Morse-smale complexes for piecewise linear 3-manifolds. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, SCG '03, 361–370, ACM, New York, NY, USA. 20

Edelsbrunner, H., Harer, J., Mascarenhas, A. & Pascucci, V. (2004a). Time-varying reeb graphs for continuous space-time data. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, 366–372, ACM, New York, NY, USA. 38

Edelsbrunner, H., Harer, J., Natarajan, V. & Pascucci, V. (2004b). Local and global comparison of continuous functions. In *Proceedings of the Conference on Visualization '04*, VIS '04, 275–280, IEEE Computer Society, Washington, DC, USA. 36

Edelsbrunner, H., Morozov, D. & Pascucci, V. (2006). Persistence-sensitive simplification functions on 2-manifolds. In *Proceedings of the Twenty-second Annual Symposium on Computational Geometry*, SCG '06, 127–134, ACM, New York, NY, USA. 25

Edelsbrunner, H., Harer, J., Mascarenhas, A., Pascucci, V. & Snoeyink, J. (2008a). Time-varying reeb graphs for continuous space–time data. *Computational Geometry*, **41**, 149 – 166. 38

Edelsbrunner, H., Harer, J. & Patel, A.K. (2008b). Reeb spaces of piecewise linear mappings. In *Proceedings of the Twenty-fourth Annual Symposium on Computational Geometry*, SCG '08, 242–250, ACM, New York, NY, USA. 30, 31

Edelsbrunner, H., Harer, J. & Patel, A.K. (2008c). Reeb Spaces of Piecewise Linear Mappings. In *Proceedings of the Twenty-fourth Annual Symposium on Computational Geometry*, SCG '08, 242–250, ACM, New York, NY, USA, event-place: College Park, MD, USA. 84, 85, 86, 87, 88, 122, 125

Emeristo. Mata-Lorenzo, L. (1986). The stein factorization for stable maps and pi-stable arcs of maps from 3-manifolds into the plane. 31

Eppstein, D. (1994). Offline algorithms for dynamic minimum spanning tree problems. *J. Algorithms*, **17**, 237–250. 23

Frattini, V., Pagnotta, S.M., Tala, Fan, J.J., Russo, M.V., Lee, S.B., Garofano, L., Zhang, J., Shi, P., Lewis, G., Sanson, H., Frederick, V., Castano, A.M., Cerulo, L., Rolland, D.C.M., Mall, R., Mokhtari, K., Elenitoba-Johnson, K.S.J., Sanson, M., Huang, X., Ceccarelli, M., Lasorella, A. & Iavarone, A. (2018). A metabolic function of fgfr3-tacc3 gene fusions in cancer. *Nature*, **553**, 222. 34

Gibbons, J., Cai, W. & Skillicorn, D.B. (1994). Efficient Parallel Algorithms for Tree Accumulations. *Science of Computer Programming*, **23**, 1 – 18. 44

Gosink, L., Anderson, J., Bethel, W. & Joy, K. (2007). Variable interactions in query-driven visualization. *IEEE Transactions on Visualization and Computer Graphics*, **13**, 1400–1407. 27

Griffith, E.J., Post, F.H., Koutek, M., Heus, T. & Jonker, H.J.J. (2005). Feature tracking in vr for cumulus cloud life-cycle studies. In *Proceedings of the 11th Eurographics Conference on Virtual Environments*, EGVE'05, 121–128, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland. 39

Gueunet, C., Fortin, P., Jomier, J. & Tierny, J. (2016). Contour Forests: Fast Multi-threaded Augmented Contour Trees. In *IEEE Symposium on Large Data Analysis and Visualization*, Baltimore, United States. 24

Gueunet, C., Fortin, P., Jomier, J. & Tierny, J. (2017). Task-based Augmented Merge Trees with Fibonacci Heaps. In *2017 IEEE 7th Symposium on Large Data Analysis and Visualization (LDAV)*, 6–15. 24

Gyulassy, A. & Natarajan, V. (2005). Topology-based simplification for feature extraction from 3d scalar fields. In *VIS 05. IEEE Visualization, 2005.*, 535–542. 25

Gyulassy, A., Natarajan, V., Pascucci, V. & Hamann, B. (2007). Efficient computation of morse-smale complexes for three-dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics*, **13**, 1440–1447. 20

Gyulassy, A., Bremer, P., Hamann, B. & Pascucci, V. (2008). A practical approach to morse-smale complex computation: Scalability and generality. *IEEE Transactions on Visualization and Computer Graphics*, **14**, 1619–1626. 20

Günther, D., Jacobson, A., Reininghaus, J., Seidel, H., Sorkine-Hornung, O. & Weinkauf, T. (2014). Fast and memory-efficienty topological denoising of 2d and 3d scalar fields. *IEEE Transactions on Visualization and Computer Graphics*, **20**, 2585–2594. 25

Hajij, M., Assiri, B. & Rosen, P. (2017). Distributed Mapper. *ArXiv e-prints*. 34

Harvey, W., Wang, Y. & Wenger, R. (2010). A randomized o(m log m) time algorithm for computing reeb graphs of arbitrary simplicial complexes. In *Proceedings of the Twenty-sixth Annual Symposium on Computational Geometry*, SoCG '10, 267–276, ACM, New York, NY, USA. 23

Heine, C., Leitte, H., Hlawitschka, M., Iuricich, F., De Floriani, L., Scheuermann, G., Hagen, H. & Garth, C. (2016). A survey of topology-based methods in visualization. *Computer Graphics Forum*, **35**, 643–667. 2

Heus, T., Jonker, H.J.J., Van den Akker, H.E.A., Griffith, E.J., Koutek, M. & Post, F.H. (2009a). A statistical approach to the life cycle analysis of cumulus clouds selected in a virtual reality environment. *Journal of Geophysical Research: Atmospheres*, **114**. 39

Heus, T., Jonker, H.J.J., Van den Akker, H.E.A., Griffith, E.J., Koutek, M. & Post, F.H. (2009b). A statistical approach to the life cycle analysis of cumulus clouds selected in a virtual reality environment. *Journal of Geophysical Research: Atmospheres*, **114**. 39

Hilaga, M., Shinagawa, Y., Kohmura, T. & Kunii, T.L. (2001a). Topology matching for fully automatic similarity estimation of 3d shapes. In *Proceedings of the*

28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '01, 203–212, ACM, New York, NY, USA. 21

Hilaga, M., Shinagawa, Y., Kohmura, T. & Kunii, T.L. (2001b). Topology matching for fully automatic similarity estimation of 3d shapes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, 203–212, ACM, New York, NY, USA. 26

Hiratuka, J.T. & Saeki, O. (2013). Triangulating Stein factorizations of generic maps and Euler characteristic formulas: Dedicated to Professor Masahiko Suzuki on the occation of his sixtieth birthday (Singularity theory, geometry and topology). *RIMS Kôkyûroku Bessatsu B38*. 84

Hristov, P. (2017). *W-Structures in Contour Trees*. Ph.D. thesis, School of Computing, University of Leeds. 25

Hristov, P. & Carr, H. (2019). W-Structures in Contour Trees. In *Proc. of TopoInVis*. 11, 24, 42, 43, 48, 60

Huettenberger, L. & Garth, C. (2015). A comparison of pareto sets and jacobi sets. In J. Bennett, F. Vivodtzev & V. Pascucci, eds., *Topological and Statistical Methods for Complex Data*, 125–141, Springer Berlin Heidelberg, Berlin, Heidelberg. 36

Huettenberger, L., Heine, C., Carr, H., Scheuermann, G. & Garth, C. (2013). Towards multifield scalar topology based on pareto optimality. In *Proceedings of the 15th Eurographics Conference on Visualization*, EuroVis '13, 341–350, The Eurographs Association., Chichester, UK. 35, 36

Huettenberger, L., Heine, C. & Garth, C. (2014). Decomposition and simplification of multivariate data using pareto sets. *IEEE Transactions on Visualization and Computer Graphics*, **20**, 2684–2693. 35

Huettenberger, L., Feige, N., Ebert, A. & Garth, C. (2015). Application of pareto sets in quality control of series production in car manufacturing. In *2015 IEEE Pacific Visualization Symposium (PacificVis)*, 135–139. 36

Huettenberger, L., Heine, C. & Garth, C. (2017a). A comparison of joint contour nets and pareto sets. In H. Carr, C. Garth & T. Weinkauf, eds., *Topological Methods in Data Analysis and Visualization IV*, 51–65, Springer International Publishing, Cham. 35

Huettenberger, L., Heine, C. & Garth, C. (2017b). A comparison of joint contour nets and pareto sets. In H. Carr, C. Garth & T. Weinkauf, eds., *Topological Methods in Data Analysis and Visualization IV*, 51–65, Springer International Publishing, Cham. 35

J. Adler, R., Agami, S. & Pranav, P. (2017). Modeling and replicating statistical topology, and evidence for cmb non-homogeneity. *Proceedings of the National Academy of Sciences*, **114**. 20

Jacobson, A., Weinkauf, T. & Sorkine, O. (2012). Smooth shape-aware functions with controlled extrema. *Computer Graphics Forum*, **31**, 1577–1586. 25

Jankowai, J. & Hotz, I. (2018a). Feature level-sets: Generalizing iso-surfaces to multi-variate data. *IEEE transactions on visualization and computer graphics*. 28, 39

Jankowai, J. & Hotz, I. (2018b). Feature Level-Sets: Generalizing Iso-surfaces to Multi-variate Data. *IEEE Transactions on Visualization and Computer Graphics*. 70

Jeitziner, R., Carrière, M., Rougemont, J., Oudot, S., Hess, K. & Brisken, C. (2018). Two-Tier Mapper: a user-independent clustering method for global gene expression analysis based on topology. *ArXiv e-prints*. 33

Kamruzzaman, M., Kalyanaraman, A. & Krishnamoorthy, B. (2018). Detecting divergent subpopulations in phenomics data using interesting flares. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, BCB '18, 155–164, ACM, New York, NY, USA. 34

Kasten, J., Hotz, I., Noack, B.R. & Hege, H.C. (2012). Vortex merge graphs in two-dimensional unsteady flow fields. *EuroVis-Short Papers*, 1–5. 39

Keller, P. & Bertram, M. (2007). Modeling and visualization of time-varying topology transitions guided by hyper reeb graph structures. In *Proceedings of the Ninth IASTED International Conference on Computer Graphics and Imaging*, CGIM '07, 15–20, ACTA Press, Anaheim, CA, USA. 38

Khambhati, A.N., Sizemore, A.E., Betzel, R.F. & Bassett, D.S. (2017). Modelling and interpreting network dynamics. *bioRxiv*. 34

Khambhati, A.N., Sizemore, A.E., Betzel, R.F. & Bassett, D.S. (2018). Modeling and interpreting mesoscale network dynamics. *NeuroImage*, **180**, 337 – 349, brain Connectivity Dynamics. 34

Klacansky, P., Tierny, J., Carr, H. & Geng, Z. (2017a). Fast and exact fiber surfaces for tetrahedral meshes. *IEEE Transactions on Visualization and Computer Graphics*, **23**, 1782–1795. 28

Klacansky, P., Tierny, J., Carr, H. & Geng, Z. (2017b). Fast and Exact Fiber Surfaces for Tetrahedral Meshes. *IEEE Transactions on Visualization and Computer Graphics*, **23**, 1782–1795. 131

Kuhn, A., Engelke, W., Flatken, M., Hege, H.C. & Hotz, I. (2017). Topology-based analysis for multimodal atmospheric data of volcano eruptions. In H. Carr, C. Garth & T. Weinkauf, eds., *Topological Methods in Data Analysis and Visualization IV*, 35–50, Springer International Publishing, Cham. 39

Landge, A.G., Pascucci, V., Gyulassy, A., Bennett, J.C., Kolla, H., Chen, J. & Bremer, P. (2014). In-situ feature extraction of large scale combustion simulations using segmented merge trees. In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 1020–1031. 23

Levine, H. (1988). Stable mappings of 3-manifolds into the plane. **20**, 279–289. 17, 84, 86, 123

Levine, H.I. (1966). Mappings of manifolds into the plane. **88**, 357–365, publisher: JSTOR. 84

Liu, X., Xie, Z. & Yi, D. (2012). A fast algorithm for constructing topological structure in large data. *Homology Homotopy Appl.*, **14**, 221–238. 34

Lorensen, W.E. & Cline, H.E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, **21**, 163–169. 10, 4, 18, 19

Lu, A. & Shen, H. (2008). Interactive storyboard for overall time-varying data visualization. In *2008 IEEE Pacific Visualization Symposium*, 143–150. 37

Lukasczyk, J., Weber, G., Maciejewski, R., Garth, C. & Leitte, H. (2017). Nested tracking graphs. *Comput. Graph. Forum*, **36**, 12–22. 38

Lukasczyk, J., Kinner, E., Ahrens, J., Leitte, H. & Garth, C. (2018). VOIDGA: A View-Approximation Oriented Image Database Generation Approach. In *8th IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, 12–22. 46, 53

Lukasczyk, J., Garth, C., Maciejewski, R. & Tierny, J. (2021). Localized topological simplification of scalar data. *IEEE Transactions on Visualization and Computer Graphics*, **27**, 572–582. 25

Luo, C., Safa, I. & Wang, Y. (2009). Approximating gradients for meshes and point clouds via diffusion metric. *Computer Graphics Forum*, **28**, 1497–1508. 30

Maadasamy, S., Doraiswamy, H. & Natarajan, V. (2012). A hybrid parallel algorithm for computing and tracking level set topology. In *2012 19th International Conference on High Performance Computing*, 1–10. 24

Manders, E.M.M., Hoebe, R., Strackee, J., Vossepoel, A.M. & Aten, J.A. (1996). Largest contour segmentation: A tool for the localization of spots in confocal images. *Cytometry*, **23**, 15–21. 26

Markov, A.A. (1958). Insolubility of the problem of homeomorphy. In *Dokl. Akad. Nauk SSSR*, vol. 121, 8. 32

Mascarenhas, A. & Snoeyink, J. (2005). Implementing time-varying contour trees. In *Proceedings of the 21st Annual Symposium on Computational Geometry, SCG'05*, 370–371. 38

Mascarenhas, A. & Snoeyink, J. (2009). *Isocontour based Visualization of Time-varying Scalar Fields*, 41–68. Springer Berlin Heidelberg, Berlin, Heidelberg. 37

Matsumoto, Y. (2002a). *An Introductino to Morse Theory*, vol. 208 of *Translation of Mathematical Monograms*. American Mathematical Society, 1st edn. 12

Matsumoto, Y. (2002b). *An introduction to Morse theory*, vol. 208. American Mathematical Soc. 25

Miettinen, K. (2012). *Nonlinear multiobjective optimization*, vol. 12. Springer Science & Business Media. 35

Miller, G.L. & Reif, J.H. (1989). Parallel Tree Contraction Part 1: Fundamentals. In S. Micali, ed., *Randomness and Computation*, 47–72, JAI Press, Greenwich, Connecticut, vol. 5. 11, 41, 44, 47

Milnor, J. (2016). *Morse Theory.(AM-51)*, vol. 51. Princeton university press. 12, 20

Moreland, K., Sewell, C., Usher, W., Lo, L.T., Meredith, J., Pugmire, D., Kress, J., Schroots, H., Ma, K.L., Childs, H., Larsen, M., Chen, C.M., Maynard, R. & Geveci, B. (2016). VTK-m: Accelerating the Visualization Toolkit for Massively Threaded Architectures. *IEEE Computer Graphics and Applications*, **36**, 48–58. 52, 57

Morozov, D. & Weber, G. (2013). Distributed merge trees. 23

Morozov, D. & Weber, G.H. (2014). Distributed contour trees. In P.T. Bremer, I. Hotz, V. Pascucci & R. Peikert, eds., *Topological Methods in Data Analysis and Visualization III*, 89–102, Springer International Publishing, Cham. 23

Munch, E. & Wang, B. (2015). Convergence between categorical representations of reeb space and mapper. *CoRR*, **abs/1512.04108**. 33

N, S. & Natarajan, V. (2011). *Simplification of Jacobi Sets*, 91–102. Springer Berlin Heidelberg, Berlin, Heidelberg. 31, 36

Nagaraj, S., Natarajan, V. & Nanjundiah, R.S. (2011). A gradient-based comparison measure for visual analysis of multifield data. *Computer Graphics Forum*, **30**, 1101–1110. 26

Newman, T.S. & Yi, H. (2006). A survey of the marching cubes algorithm. *Computers & Graphics*, **30**, 854–879. 19

Ni, X., Garland, M. & Hart, J.C. (2004). Fair morse functions for extracting the topological structure of a surface mesh. *ACM Trans. Graph.*, **23**, 613–622. 25

Nicolau, M., Levine, A.J. & Carlsson, G. (2011). Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, **108**, 7265–7270. 34

Nielson, J.L., Paquette, J., Liu, A.W., Guandique, C.F., Tovar, C.A., Inoue, T., Irvine, K.A., Gensel, J.C., Kloke, J., Petrossian, T.C., Lum, P.Y., Carlsson, G.E., Manley, G.T., Young, W., Beattie, M.S., Bresnahan, J.C. & Ferguson, A.R. (2015). Topological data analysis for discovery in preclinical spinal cord injury and traumatic brain injury. *Nature Communications*, **6**, 8581 EP –, article. 34

Norgard, G. & Bremer, P.T. (2013). Ridge–valley graphs: Combinatorial ridge detection using jacobi sets. *Computer Aided Geometric Design*, **30**, 597 – 608, foundations of Topological Analysis. 31

Norton, A. & Clyne, J. (2012). The vapor visualization application. *High Performance Visualization*, 415–428. 39

Oesterling, P., Heine, C., Weber, G.H., Morozov, D. & Scheuermann, G. (2017). Computing and visualizing time-varying merge trees for high-dimensional data. In H. Carr, C. Garth & T. Weinkauf, eds., *Topological Methods in Data Analysis and Visualization IV*, 87–101, Springer International Publishing, Cham. 38

O'Leary, P., Ahrens, J., Jourdain, S., Wittenburg, S., Rogers, D.H. & Petersen, M. (2016). Cinema Image-based in situ Analysis and Visualization of MPAS-ocean Simulations. *Parallel Computing*, **55**, 43 – 48, visualization and Data Analytics for Scientific Discovery. 46

Orf, L.G., Semeraro, B.D. & Wilhelmson, R.B. (2007). Vortex detection in a simulated supercell thunderstorm. *Atmospheric Science Letters*, **8**, 29–35. 39

Parsa, S. (2013). A deterministic $o(mlog(m)$ time algorithm for the reeb graph. *Discrete & Computational Geometry*, **49**, 864–878. 23

Pascucci, V. & Cole-McLaughlin, K. (2004). Parallel computation of the topology of level sets. *Algorithmica*, **38**, 249–268. 23

Pascucci, V., Cole-McLaughlin, K. & Scorzell, G. (2004a). Multi-Resolution Computation and Presentation of Contour Trees. In *Proceedings of the IASTED conference on Visualization, Imaging and Image Processing (VIIP 2004)*, 452–290. 12, 43, 48, 60, 61, 130

Pascucci, V., Cole-McLaughlin, K. & Scorzelli, G. (2004b). Multi-resolution computation and presentation of contour trees. In *Proc. IASTED Conference on Visualization, Imaging, and Image Processing*, 452–290, Citeseer. 25

Pascucci, V., Scorzelli, G., Bremer, P.T. & Mascarenhas, A. (2007). Robust on-line computation of reeb graphs: Simplicity and speed. *ACM Trans. Graph.*, **26**. 22, 26, 37

Phinyomark, A., Petri, G., Ibáñez-Marcelo, E., Osis, S.T. & Ferber, R. (2018). Analysis of big data in gait biomechanics: Current trends and future directions. *Journal of Medical and Biological Engineering*, **38**, 244–260. 34

Post, F.H., Vrolijk, B., Hauser, H., Laramee, R.S. & Doleisch, H. (2004). The state of the art in flow visualisation: Feature extraction and tracking. *Computer Graphics Forum*, **22**, 775–792. 37, 39

Rautenhaus, M., Böttinger, M., Siemen, S., Hoffman, R., Kirby, R.M., Mirzargar, M., Röber, N. & Westermann, R. (2018). Visualization in meteorology—a survey of techniques and tools for data analysis tasks. *IEEE Transactions on Visualization and Computer Graphics*, **24**, 3268–3296. 39

Reeb, G. (1946). Sur les points singuliers d'une forme de pfaff completement integrable ou d'une fonction numerique [on the singular points of a completely integrable pfaff form or of a numerical function]. *Comptes Rendus Acad. Sciences Paris*, **222**, 847–849. 20

Reimann, M.W., Nolte, M., Scolamiero, M., Turner, K., Perin, R., Chindemi, G., Dłotko, P., Levi, R., Hess, K. & Markram, H. (2017). Cliques of neurons bound into cavities provide a missing link between structure and function. *Frontiers in Computational Neuroscience*, **11**, 48. 20

Reininghaus, J., Kasten, J., Weinkauf, T. & Hotz, I. (2012). Efficient computation of combinatorial feature flow fields. *IEEE Transactions on Visualization and Computer Graphics*, **18**, 1563–1573. 38

Rizvi, A.H., Camara, P.G., Kandror, E.K., Roberts, T.J., Schieren, I., Maniatis, T. & Rabadan, R. (2017). Single-cell topological rna-seq analysis reveals insights into cellular differentiation and development. *Nature biotechnology*, **35**, 551. 34

Robins, V., Wood, P.J. & Sheppard, A.P. (2011). Theory and algorithms for constructing discrete morse complexes from grayscale digital images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **33**, 1646–1658. 20

Rosen, P., Tu, J. & Piegl, L.A. (2018). A Hybrid Solution to Parallel Calculation of Augmented Join Trees of Scalar Fields in any Dimension. *Computer-Aided Design and Applications*, **15**, 610–618. 24

Rossi, P. (1840). *Cours d'économie politique*, vol. 1. Société typographique belge. 35

Rourke, C.P. & Sanderson, B. (1972). *Introduction to Piecewise-Linear Topology*. Springer-Verlag, Berlin; Heidelberg; New York. 15

Saeki, O. (2004). *Topology of singular fibers of differentiable maps*. Springer. 14, 84, 85

Saeki, O. (2017). Theory of Singular Fibers and Reeb Spaces for Visualization. In H. Carr, C. Garth & T. Weinkauf, eds., *Topological Methods in Data Analysis and Visualization IV*, 3–33, Springer International Publishing, Cham. 14, 84, 126

Saeki, O. & Yamamoto, T. (2016a). Cobordism group of morse functions on surfaces with boundary. In *XIII International Workshop, Real and Complex Singularities, Universidade de Sao Paulo; Contemporary Mathematics*, vol. 675, 279–297. 84

Saeki, O. & Yamamoto, T. (2016b). Singular fibers of stable maps of 3–manifolds with boundary into surfaces and their applications. *Algebraic & Geometric Topology*, **16**, 1379–1402, publisher: Mathematical Sciences Publishers. 84

Saggar, M., Sporns, O., Gonzalez-Castillo, J., Bandettini, P.A., Carlsson, G., Glover, G. & Reiss, A.L. (2018). Towards a new approach to reveal dynamical organization of the brain using topological data analysis. *Nature Communications*, **9**, 1399. 34

Sakurai, D., Saeki, O., Carr, H., Wu, H., Yamamoto, T., Duke, D. & Takahashi, S. (2016). Interactive visualization for singular fibers of functions. *IEEE Transactions on Visualization and Computer Graphics*, **22**, 945–954. 29, 30

Samtaney, R., Silver, D., Zabusky, N. & Cao, J. (1994). Visualizing features and tracking their evolution. *Computer*, **27**, 20–27. 37

Sauber, N., Theisel, H. & Seidel, H. (2006). Multifield-graphs: An approach to visualizing correlations in multifield scalar data. *IEEE Transactions on Visualization and Computer Graphics*, **12**, 917–924. 27

Schneider, D., Wiebel, A., Carr, H., Hlawitschka, M. & Scheuermann, G. (2008a). Interactive comparison of scalar fields based on largest contours with applications to flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, **14**, 1475–1482. 26

Schneider, D., Wiebel, A., Carr, H., Hlawitschka, M. & Scheuermann, G. (2008b). Interactive Comparison of Scalar Fields Based on Largest Contours with Applications to Flow Visualization. *IEEE Transactions on Visualization and Computer Graphics*, **14**, 1475–1482. 46

Scoville, N.A. (2019). *Discrete Morse Theory*, vol. 90. American Mathematical Soc. 20

Shinagawa, Y. & Kunii, T.L. (1991). Constructing a reeb graph automatically from cross sections. *IEEE Computer Graphics and Applications*, **11**, 44–51. 21

Shinagawa, Y., Kunii, T.L. & Kergosien, Y.L. (1991). Surface coding based on morse theory. *IEEE Computer Graphics and Applications*, **11**, 66–78. 20

Shirley, P. & Tuchman, A. (1990). A polygonal approximation to direct scalar volume rendering. *SIGGRAPH Comput. Graph.*, **24**, 63–70. 19

Singh, G., Memoli, F. & Carlsson, G. (2007). Topological methods for the analysis of high dimensional data sets and 3d object recognition. In M. Botsch, R. Pajarola, B. Chen & M. Zwicker, eds., *Eurographics Symposium on Point-Based Graphics*, The Eurographics Association. 32

Snyder, D.F. (2004). Topological persistence in jacobi sets. Tech. rep., Technical report, Technical Report July 29. 36

Sohn, B.. & Bajaj, C. (2006). Time-varying contour topology. *IEEE Transactions on Visualization and Computer Graphics*, **12**, 14–25. 37

Soler, M., Plainchault, M., Conche, B. & Tierny, J. (2018). Lifted Wasserstein Matcher for Fast and Robust Topology Tracking. *ArXiv e-prints*. 38

Sorgente, T., Biasotti, S., Livesu, M. & Spagnuolo, M. (2018). Topology-driven shape chartification. *Computer Aided Geometric Design*, **65**, 13 – 28. 21

Spivak, M. (2018). *Calculus on manifolds: a modern approach to classical theorems of advanced calculus*. CRC press. 9

Steiner, D. & Fischer, A. (2001). Topology recognition of 3d closed freeform objects based on topological graphs. In *Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications*, SMA '01, 305–306, ACM, New York, NY, USA. 21

Stevens, B., Moeng, C.H. & Sullivan, P.P. (1999). Large-eddy simulations of radiatively driven convection: Sensitivities to the representation of small scales. *Journal of the Atmospheric Sciences*, **56**, 3963–3984. 78

Strodthoff, B. & Jüttler, B. (2013). Layered reeb graphs of a spatial domain. 26

Strodthoff, B. & Jüttler, B. (2015). Layered reeb graphs for three-dimensional manifolds in boundary representation. *Computers & Graphics*, **46**, 186 – 197, shape Modeling International 2014. 26

Szymczak, A. (2005). Subdomain aware contour trees and contour evolution in time-dependent scalar fields. In *International Conference on Shape Modeling and Applications 2005 (SMI' 05)*, 136–144. 37

Tarjan, R.E. (1975). Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, **22**, 215–225. 23

Thomas, D.P., Borgo, R., Carr, H. & Hands, S. (2017). Joint contour net analysis of lattice qcd data. *arXiv preprint arXiv:1703.02488*. 34, 35

Tierny, J. & Carr, H. (2017a). Jacobi fiber surfaces for bivariate reeb space computation. *IEEE Transactions on Visualization and Computer Graphics*, **23**, 960–969. 32, 87

Tierny, J. & Carr, H. (2017b). Jacobi Fiber Surfaces for Bivariate Reeb Space Computation. *IEEE Transactions on Visualization and Computer Graphics*, **23**, 960–969. 85

Tierny, J. & Pascucci, V. (2012). Generalized topological simplification of scalar fields on surfaces. *IEEE Transactions on Visualization & Computer Graphics*, **18**, 2005–2013. 25

Tierny, J., Gyulassy, A., Simon, E. & Pascucci, V. (2009). Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees. *IEEE Transactions on Visualization and Computer Graphics*, **15**, 1177–1184. 22

Tierny, J., Favelier, G., Levine, J.A., Gueunet, C. & Michaux, M. (2017). The Topology ToolKit. *IEEE Transactions on Visualization and Computer Graphics (Proc. of IEEE VIS)*, https://topology-tool-kit.github.io/. 46, 53

Tierny, J., Favelier, G., Levine, J.A., Gueunet, C. & Michaux, M. (2018). The topology toolkit. *IEEE Transactions on Visualization and Computer Graphics*, **24**, 832–842. 29

Tominski, C., Donges, J.F. & Nocke, T. (2011). Information visualization in climate research. In *Information Visualisation (IV), 2011 15th International Conference on*, 298–305, IEEE. 39

Treece, G., Prager, R. & Gee, A. (1999a). Regularised marching tetrahedra: improved iso-surface extraction. *Computers & Graphics*, **23**, 583 – 598. 19

Treece, G.M., Prager, R.W. & Gee, A.H. (1999b). Regularised Marching Tetrahedra: Improved Iso-surface Extraction. *Computers And Graphics*, **23**, 583–598. 88, 89

van Gelder, A. & Wilhelms, J. (1994). Topological considerations in isosurface generation. *ACM Trans. Graph.*, **13**, 337–375. 19

vanZanten, M.C., Stevens, B., Nuijens, L., Siebesma, A.P., Ackerman, A., Burnet, F., Cheng, A., Couvreux, F., Jiang, H., Khairoutdinov, M. *et al.* (2011). Controls on precipitation and cloudiness in simulations of trade-wind cumulus as observed during RICO. *Journal of Advances in Modeling Earth Systems*, **3**. 78

Verovšek, S.K. & Mashaghi, A. (2016). Extended topological persistence and contact arrangements in folded linear molecules. *Frontiers in Applied Mathematics and Statistics*, **2**, 6. 20

Wang, C., Yu, H. & Ma, K. (2008). Importance-driven time-varying data visualization. *IEEE Transactions on Visualization and Computer Graphics*, **14**, 1547–1554. 37

Weber, G., Dillard, S., Carr, H., Pascucci, V. & Hamann, B. (2007a). Topology-Controlled Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, **13**, 330–341. 50

Weber, G.H., Dillard, S.E., Carr, H., Pascucci, V. & Hamann, B. (2007b). Topology-controlled volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, **13**, 330–341. 21

Weinkauf, T., Gingold, Y. & Sorkine, O. (2010). Topology-based smoothing of 2d scalar fields with c1-continuity. In *Proceedings of the 12th Eurographics / IEEE - VGTC Conference on Visualization*, EuroVis'10, 1221–1230, The Eurographs Association., Chichester, UK. 25

Wenger, R. (2013a). *Isosurfaces: geometry, topology, and algorithms*. AK Peters/CRC Press. 18

Wenger, R. (2013b). *Isosurfaces: geometry, topology, and algorithms*. AK Peters/CRC Press. 70

Whitney, H. (1955). On singularities of mappings of euclidean spaces. i. mappings of the plane into the plane. In J. Eells & D. Toledo, eds., *Hassler Whitney Collected Papers*, 370–406, Birkhäuser Boston. 84

Widanagamaachchi, W., Christensen, C., Pascucci, V. & Bremer, P. (2012). Interactive exploration of large-scale time-varying data using dynamic tracking graphs. In *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, 9–17. 37

Wood, Z., Hoppe, H., Desbrun, M. & Schröder, P. (2004). Removing excess topology from isosurfaces. *ACM Trans. Graph.*, **23**, 190–208. 21

Wood, Z.J., Desbrun, M., Schroder, P. & Breen, D. (2000). Semi-regular mesh extraction from volumes. In *Proceedings Visualization 2000. VIS 2000 (Cat. No.00CH37145)*, 275–282. 21

Wu, K., Knoll, A., Isaac, B.J., Carr, H. & Pascucci, V. (2017). Direct multifield volume ray casting of fiber surfaces. *IEEE Transactions on Visualization and Computer Graphics*, **23**, 941–949. 29

Wyvill, G., McPheeters, C. & Wyvill, B. (1986). Data Structure for Soft Objects. *Visual Computer*, **2**, 227–234. 50

Yao, Y., Sun, J., Huang, X., Bowman, G.R., Singh, G., Lesnick, M., Guibas, L.J., Pande, V.S. & Carlsson, G. (2009). Topological methods for exploring low-density states in biomolecular folding pathways. *The Journal of Chemical Physics*, **130**, 144115. 34

Yu, L., Lu, A. & Chen, W. (2013). Visualization and analysis of 3d time-varying simulations with time lines. *Journal of Visual Languages & Computing*, **24**, 402 – 418. 37

Zhang, E., Mischaikow, K. & Turk, G. (2005). Feature-based surface parameterization and texture mapping. *ACM Trans. Graph.*, **24**, 1–27. 21

Zhang, X., Marcos, Bajaj, C.L. & Baker, N. (2004). Fast matching of volumetric functions using multi-resolution dual contour trees. 26

Èliašberg, J.M. (1970). ON SINGULARITIES OF FOLDING TYPE. *Mathematics of the USSR-Izvestiya*, **4**, 1119–1134, publisher: IOP Publishing. 84