# A Topological Framework for Visualizing Time-varying Volumetric Datasets

Ajith Mascarenhas
Department of Computer Science
University of North Carolina at Chapel Hill

May 18, 2004

**Abstract**

Thesis statement: The *Time-varying Reeb Graph* provides a topological framework to perform visualization of a time-varying volumetric dataset. It assists us to compute the number and genus of level-set components for all (function value, time) pairs, compute seed-cells for fast level-set extraction, and perform temporal simplification of level-set topology.

## 1 Introduction

Physical processes that are measured over time, or that are modeled and simulated on a computer, can produce large amounts of time-varying data that must be interpreted with the assistance of computational tools. Such data arises in a wide variety of studies including computational fluid dynamics, oceanography, climate modeling. Often the data is scalar valued, with perhaps several scalar values computed for each sample point. E.g: Pressure, temperature, density. A study of motion or velocity of some kind will result in vector-valued data. Since the data is sampled regularly, semi-regularly, or irregularly in both time and space, interpolation allows us to consider it as a continuous scalar or vector field. Piecewise-linear interpolation is common for large amounts of data, because of its relative ease; multilinear interpolation is also used for regular grids.

The goal of simulation is to gain insight into the process studied and this is often achieved by interpreting the data by *transforming* it into a visual form. Images and movies of the transformed data are easier to understand to a trained scientist than a large amount of numbers and hence techniques and tools to visualize data are valuable.

A useful tool to interpret the data is graphical visualization, often by computing slices or level-sets (also called isosurfaces) of the data. In slicing, we restrict the data which lies in 3D or higher to a suitable plane, and display this restriction on a computer screen. By varying the plane of slicing we can study the variation of data in space and time. In level-set based visualization, we fix a scalar value, compute the points in space with that value and display the results. By varying the scalar value we can study the variation in the data. Both these methods are amenable to the visualization of scalar valued data-sets. Vector-valued data-sets are typically visualized either component-wise, or by the use of a transfer function (e.g: magnitude of velocity) to produce a scalar valued data-set, and then using either slicing or level-set visualization. In this thesis, we will focus on scalar-valued data-sets and on tools to understand level-set behavior.

We address the following question when visualizing using level-sets: what level values result in interesting level-sets that provide insight into the process? Moreover, the evolution of the level-set as the level value is continuously varied is also important. How do the different components of a level-set interact? At what level values do new components appear, disappear, merge, split or change genus? What tools exist that can provide this topological information without actually computing each possible level-set? Since time-varying volume data gives a large space of slice and level-set parameters to explore, development of tools to intuitively present the topological properties of the level-sets for all level values and time, is an important step towards improved visualization.

I propose the time-varying Reeb graph as a topological framework to perform visualization of time-varying volumetric datasets. Intuitively, the Reeb graph encodes topological changes to the level set as the level value is varied. These changes are creation, destruction, split, merge and genus change of level set components. The Reeb graph can also be used to extract level-sets efficiently, by finding a seed edge for each connected component of the level-set, and visiting only those mesh elements that intersect the level-set.

In the case of time-varying data, we are interested in the Reeb graph of the volume slice at every instant of time. We raise the following questions:

- How can we compute the Reeb graph for every instant of time and store it compactly in a time-varying Reeb graph?
- How do we augment the time-varying Reeb graph with information about the topology of the level-sets?
- How can we use time-varying Reeb graphs to identify ”short-lived” level-set components and remove them?

In the following sections, we lay out a research plan aimed at answering these questions. The document is organized as follows: In section 2 we provide the necessary background, in section 3 we discuss related prior work, in section 4 we describe the theory and algorithm developed to compute time-varying Reeb graphs and augment them with topological information about the level-sets, and in section 5 we discuss possible research directions to perform temporal simplification using time-varying Reeb graphs. Section 6 lists expected results, section 7 provides a classification of data-sets available, section 8 lists contributions by coauthors, and section 9 outlines the progress to date and planned milestones.

# 2 Background

Our work draws from and extends concepts from Morse theory [20][21] and from combinatorial topology [22]. We provide some background on concepts used.

**Smooth maps on manifolds.** Let $\mathbb{M}$ be a smooth, compact $d$-manifold without boundary, and $f\colon \mathbb{M} \to \mathbb{R}$ a smooth map. Assuming a local coordinate system in its neighborhood, a point $x \in \mathbb{M}$ is a *critical point* of $f$ if all derivatives of $f$ vanish at $x$, $\frac{\partial f}{\partial x_i} = 0$. If $x$ is a critical point, $f(x)$ is a *critical value*. Non-critical points and non-critical values are called *regular points* and *regular values*, respectively. The *Hessian* at $x$ is the matrix of second order partial derivatives,

$$H(x) = \left( \frac{\partial^2 f}{\partial x_i \partial x_j}(x) \right)$$

A critical point $x$ is *non-degenerate* if $H(x)$ is non-singular. The *index* of a critical point $x$, denoted $\text{index } x$, is the number of negative eigenvalues of $H(x)$. Intuitively, it is the number of mutually orthogonal directions at $x$ along which $f$ decreases. For $d = 3$, there are four types of non-degenerate critical points: *minima* have index 0, 1-*saddles* have index 1, 2-*saddles* have index 2, and *maxima* have index 3. A function $f$ is a *Morse function* if

I. all its critical points are non-degenerate, and

II. $f(x) \neq f(y)$ whenever $x \neq y$ are critical.

We will refer to I and II as Genericity Conditions as they prevent certain non-generic configurations of the critical points. This choice of name is justified because Morse functions are dense in $C^\infty(\mathbb{M})$, the class of smooth functions on the manifold. In other words, for every smooth function there is an arbitrarily small perturbation that makes it a Morse function.

The Euler characteristic of $\mathbb{M}$ expressed in terms of the critical points $x$ of $f$ is $\chi(\mathbb{M}) = \sum_x (-1)^{\text{index } x}$.

A *level set* is the preimage of a constant value, $f^{-1}(s)$, $s \in \mathbb{R}$. If $s$ is a regular value then $f^{-1}(s)$ is a $(d-1)$-manifold.

**Reeb Graph.** A level set of $f$ is not necessarily connected. If we call two points $x, y \in \mathbb{M}$ *equivalent* when $f(x) = f(y)$ and both points belong to the same component of the level set, then we obtain the *Reeb graph* as the quotient space in which every equivalence class is represented by a point and connectivity is defined in terms of the quotient topology [24].

We call a point on the Reeb graph a *node* if the corresponding level set passes through a critical point of $f$. The rest of the Reeb graph consists of arcs connecting the nodes. The *degree* of a node is the number of arcs that are incident to the node. A minimum creates and a maximum destroys a level set component and thus correspond to degree-1 nodes. A saddle point splits or merges components. If a saddle either splits one component into two or merges two components into one, then the corresponding node is of degree-3. If a saddle does not change the number of components, then it corresponds to a degree-2 node and the level set changes genus only. Nodes of degree higher than three occur only for non-Morse functions.

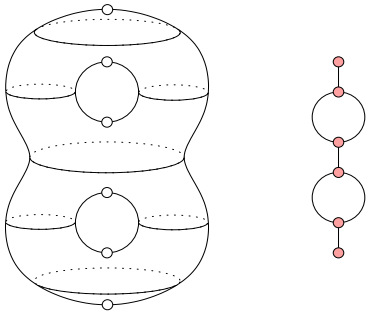

Figure 1: The Reeb graph of the function $f$ on a 2-manifold, which maps every point of the double torus to its distance above a horizontal plane below the surface.

In mathematics, the Reeb graph is often used to study the manifold $\mathbb{M}$ that forms the domain of the function. For example, the Reeb graph in Figure 1 reveals that the function is defined on a double torus, assuming we know it is an orientable 2-manifold

without boundary. In contrast, we will use the Reeb graph to study the behavior of functions. The domain of interest is $\mathbb{R}^3$ but it is convenient to compactify it and consider functions on the 3-sphere, $\mathbb{S}^3$. All our Reeb graphs will reveal the (un-exciting) connectivity of $\mathbb{S}^3$ by being trees, but the structure of the tree will tell us something about the chosen function $f$. Thus, aspects of the literature that focus on Reeb graphs in $\mathbb{S}^3$ or $\mathbb{R}^3$ sometime call them *contour trees* [6].

**Jacobi curves.** We use Reeb graphs to understand a function at moments in time and Jacobi curves, as introduced in [12], to get a glimpse of its evolution through time. We introduce this concept for the slightly more general case of two Morse functions, $f, g : \mathbb{M} \to \mathbb{R}$; the specific case of a time-varying function, $f$, is obtained by adding time as an extra dimension to the domain and letting $g$ represent time. For a regular value $t \in \mathbb{R}$, we have the level set $g^{-1}(t)$ and the restriction of $f$ to this level set, $f_t : g^{-1}(t) \to \mathbb{R}$. The *Jacobi curve* of $f$ and $g$ is the closure of the set of critical points of the functions $f_t$, for all $t \in \mathbb{R}$. The closure operation adds the critical points of $f$ restricted to level sets at critical values, as well as the critical points of $g$, which form singularities in these level sets.

We consider a 1-parameter family of Morse functions on the 3-sphere, $f : \mathbb{S}^3 \times \mathbb{R} \to \mathbb{R}$, and introduce an auxiliary function $g : \mathbb{S}^3 \times \mathbb{R} \to \mathbb{R}$ defined by $g(x, t) = t$. A level set has the form $g^{-1}(t) = \mathbb{S}^3 \times t$, and the restriction of $f$ to this level set is $f_t : \mathbb{S}^3 \times t \to \mathbb{R}$. The Jacobi curve of $f$ and $g$ may consist of several components, and in the assumed generic case each is a closed 1-manifold. We can identify the *birth-death points* where the level sets of $f$ and $g$ and the Jacobi curve have a common normal direction. To understand these points, imagine a level set in the form of a (two-dimensional) sphere deforming, sprouting a bud, as we go forward in time. The bud has two critical points, one a maximum and the other a 2-saddle. At the time when the bud just starts sprouting there is a point on the sphere, a birth point, where both these critical points are born. Run this in reverse order to understand a death point. Birth-death points suggest a further decomposition of each component into *segments* where each segment is the trace of a critical point of $f_t$ between its birth and death point. The index of the critical point tracing a segment is the same everywhere along the segment.

**Betti numbers** The *k-th Betti number*, denoted as $\beta_k$, is the rank of the $k$-th homology group: $\beta_k = \operatorname{rank} \mathsf{H}_k$. Intuitively, $\beta_0$ is the number of connected components of the space, $\beta_1$ is the number of tunnels, and $\beta_2$ is the number of voids of the space. For example a 2-torus $T^2$ has the following Betti numbers: $\beta_0 = 1$, $\beta_1 = 2$, and $\beta_2 = 1$. The torus has two cycles, one latitudinal and one longitudinal cycle bounding a tunnel each, hence $\beta_1 = 2$.

# 3 Related Prior Work

In this section we look at prior work related to visualization of volumetric datasets. Most of the work done so far has concentrated on static 3D volumes. The data is usually given in the form of a collection of scalar values at regularly or irregularly sampled points in space. Often the space is decomposed into cubical *cells* in the case of regular grids or simplices in the case of irregularly sampled data. Interpolation is used to get a continuous scalar valued function throughout the space.

**Iso-surface extraction** There has been a considerable body of work developed in isosurface extraction, of which we look at a selection here. The *Marching Cubes* algorithm [18], enumerates all cells in the volume and computes the part of the isosurface intersecting each cell based on a classification of each vertex of the cell. A vertex gets a *positive* label if its function value is greater than the iso-value and a *negative* label otherwise. The drawback with this method is that the ratio of the cells inspected to the cells actually intersecting the isosurface is large. Wilhelms and van Gelder [30] improve the search by using an octree spatial decomposition to skip regions that do not intersect the isosurface.

Using another approach, [15][14][19] process the cells of a mesh based on their *value space*, rather than the *domain space*. Giles and Haines [15] form two sorted lists of cells, one by maximum value and the other by minimum value. Using the overall maximum cell range $\Delta w$, they limit the search to cells with minimum value in the range $[w - \Delta w, w]$, for a given isovalue $w$. Livnat et al.[19] use a *kd-tree* to simultaneously order cells according to their minimum and maximum values, allowing them an $O(\sqrt{n} + k)$ extraction time, where $n$ is the mesh size and $k$ is the output size. A drawback of these search techniques is the significant memory overhead to construct and maintain the associated data-structures.

**Seed-set computation**

*Definiton:* A *seed-set* is a subset of cells of the mesh, such that the set contains at least one cell intersecting each connected component of each isosurface.

Seed set based methods [2][28] compute a seed-set and construct a search structure on the significantly smaller sized seed set. Contour propagation (also called mesh propagation) [16][17] is used to extract the isosurface, starting from seed cells. van Kreveld et al. [28] describe construction of a minimal seed-set using the contour tree in polynomial time. They also provide a practical approximation algorithm that runs in $O(nlog^2n)$ time and linear space in 2D, and $O(n^2)$ time and linear space in higher dimensions. Pascucci et al. [4] present three algorithms for generating seed sets, two applicable to regular and irregular grids of arbitrary dimension, and one specialized to regular grids. Recently, Carr [7] has developed the notion of a *path seed*. An edge in the mesh that intersects a level-set can be used as a seed for that level-set. Carr chooses seed edges such that they form

monotone paths starting at critical points. By storing information to start tracing these monotone paths at nodes in the contour tree, seed edges can be computed whenever required, by tracing monotone paths starting from appropriate critical points.

**Reeb Graphs, Contour Trees** The Reeb graph of a function defined on a 3D volume is a tree, and is also referred to as the *Contour Tree*[6]. van Kreveld et al. [28] describe an algorithm to compute contour trees for 2D meshes with $n$ elements in $O(nlog(n))$ time and $O(n^2)$ time in higher dimensions. Their algorithm sweeps through the data from low to high values, while maintaining each level-set component. On encountering a critical point, they determine its type, and merge or split the affected level sets. Tarasov and Vyalyi [27] presented an $O(nlog(n))$ algorithm for 3D meshes. Their algorithm performs three sweeps: the first to identify joins, the second in the reverse direction to identify splits, and the third to combine the results of the previous two sweeps. Carr et al. [6] presented an improved algorithm that computes the contour tree in arbitrary but constant dimension in time $O(n \log n + N\alpha(N))$ where the the mesh has $n$ vertices, $N$ simplices, and $\alpha(.)$ is the very slowly growing inverse of ackerman's function. For the case of 3-manifolds, this algorithm has been extended to include information about the genus of the level sets in [23]. Cole-McLaughlin et al. [8] provide tight upper and lower bounds on the number of loops in the Reeb graph of a Morse function defined on a 2-manifold, and provide an $O(n \log n)$ time algorithm to construct the Reeb graph, where $n$ is the number of edges in the triangulation of the the 2-manifold.

**The Contour Spectrum**

*Definition:* The *Contour Spectrum* is a collection of graphs of a variety of scalar data and contour attributes (e.g. contour area, volume), computed over the range of scalar values.

Bajaj et al. propose the use of the Contour Spectrum [3] to guide selection of the level during visualization. Their interface provides graphs for level-set properties as a function of the level. The properties graphed include the surface area and enclosed volume of the level-set. They showed that surface area and enclosed volume could be computed and displayed without computing all of the isosurfaces. This is because these properties are decomposable: their values over the entire surface can be obtained by combining values computed in each interpolation cell. Within each interpolation cell, they show that the properties are given by a univariate B-spline function of the level. The sum of these splines results in a global spline function for the property.

**Visualization of time-varying volume datasets** Weigle et al.[29] extract 2D isosurfaces from time-varying data in 4D. They first extract a 3D solid mesh with all function values equal to the iso-value, and then extract a 2D isosurface from this mesh using time as the iso-value. Their method is able to generate smooth animations, but is time consuming. For a mesh with $40 \times 40 \times 40 \times 36$ points they report an isosurface extraction time on the order of 2 minutes. The Temporal Hierarchical Index (THI) Tree proposed by Shen [25] adaptively coalesces cells, based on temporal variation, and stores them in a tree structure. This results in a space saving for the search structure, but is still prohibitively large to completely fit in main memory. Sutton et al. [26] propose the Temporal Branch-on-Need Tree (T-BON) to minimize unnecessary I/O access by constructing the octree decomposition of Wilhelms and van Gelder[30] for each time step. Bajaj et al. [5] perform progressive tracking of isosurfaces in time-varying scalar fields by using temporal contour propagation of the isosurface from time $t$ to compute the isosurface at time $t + 1$. New isosurface components created in time $t + 1$ are generated from seed sets for that time step.

# 4 Time-varying Reeb Graphs: Theory and Algorithms

This section explains briefly some of the work done towards supporting my thesis. We show that it is possible to classify the changes in the Reeb graph $R_t$ of $f_t$ as we vary time $t$. We provide an algorithm to perform these changes and compute the time-varying Reeb graph.

**Jacobi curves connect Reeb graphs.** Let $R_t$ be the Reeb graph of $f_t$, the function on $\mathbb{S}^3$ at time $t$. The nodes of $R_t$ correspond to critical points of $f_t$, and as we vary $t$, they trace out the segments of the Jacobi curve. The segments connect the family through time, giving us a mechanism for identifying nodes in different Reeb graphs. We illustrate this idea in Figure 2.

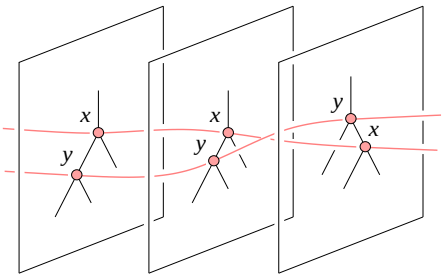


Figure 2:Reeb graphs at three moments in time whose nodes are connected by two segments of the Jacobi curve.

Generically, the function $f_t$ is Morse. However, there are discrete moments in time at which $f_t$ violates one or both Genericity Conditions of Morse functions and the Reeb graph of $f_t$ experiences a combinatorial change. Since we have only one varying parameter, namely time, we may assume that there is only a single violation of the Genericity Conditions at any of these discrete moments, and there are no violations at all other times. Condition I is violated iff $f_t$ has a birth-death point at which a cancellation annihilates two converging critical points or an anti-cancellation gives birth to two diverging critical points. Condition II is violated iff $f_t$ has two critical points $x \neq y$ with $f_t(x) = f_t(y)$ that form an interchange. The two critical points may be independent and have no effect on the Reeb graph, or they may belong to the same level set component of $f_t$ and correspond to two nodes that swap their positions along the Reeb graph. We provide a brief description of the changes caused by birth-death points and by interchanges, for details see [13]

**Nodes appear and disappear.** A pair of critical points are created at a birth point of the Jacobi curve; the corresponding nodes, joined by an arc, appear in the Reeb graph $R_t$. We can have three cases depending on the indices of the critical point pairs that are created: $(0, 1)$ corresponding to a minimum, index-1 pair, $(1, 2)$ corresponding to a index-1, index-2 pair, and $(2, 3)$ corresponding to a index-2, maximum pair. See Figure 3.

A pair of critical points annihilate each other at a death point; in $R_t$, an arc joining corresponding nodes contracts and the nodes disappear. This is the reverse of node appearance, so we have the same cases discussed above, and illustrated in Figure 3 if we read it from right to left.

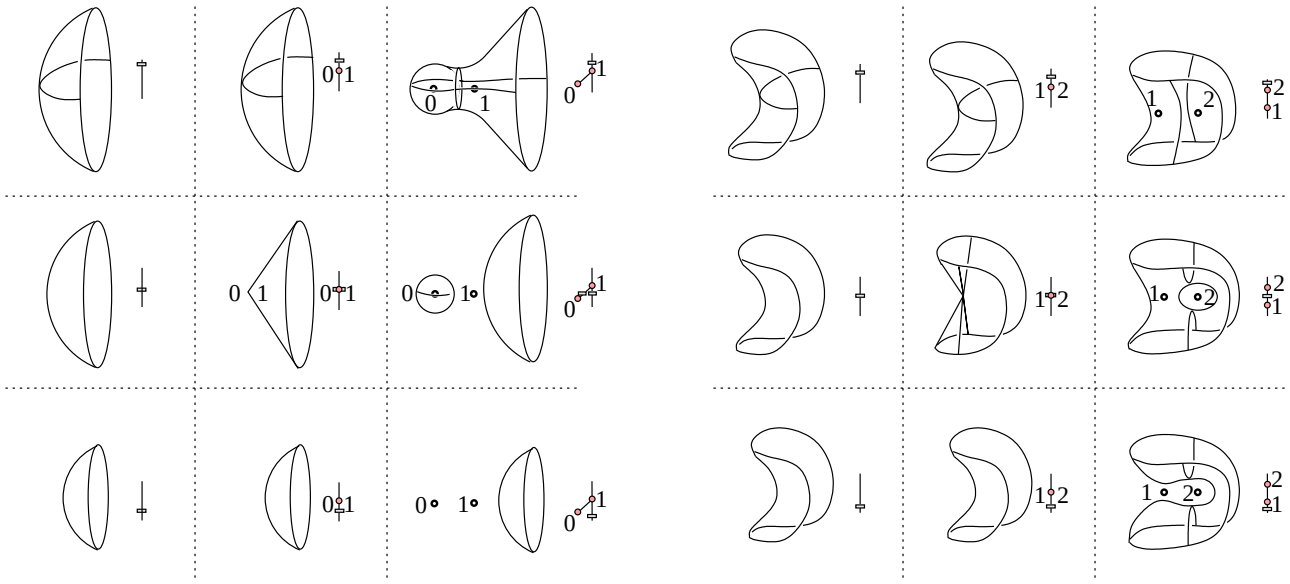


Figure 3: Level sets and Reeb graphs around a 0-1 birth point on the left, and a 1-2 birth point on the right. Time increases from left to right and the level set parameter, indicated by a rectangular slider bar, increases from bottom to top. Going forward in time, we see the sprouting of a bud, while going backward in time we see its retraction.

**Node swap.** A pair of critical points $x$ and $y$, with index 1 or 2, and whose corresponding nodes are joined by an arc in the Reeb graph, may change order along $f$ at time $t$. Without loss of generality, we may assume that $f_{t-\varepsilon}(x) < f_{t-\varepsilon}(y)$ and $f_{t+\varepsilon}(x) > f_{t+\varepsilon}(y)$. We have four choices for each of $x$ and $y$ depending on whether they add or remove a handle, merge two level set components or split a level set component. This gives a total of sixteen configurations. We analyze possible before and after combinations and pair them, giving us the cases illustrated in Figure 4. It is convenient to group the cases with similar starting configurations together. We use $+, -, \mathrm{M}, \mathrm{S}$ to mean 'handle addition', 'handle deletion', 'component merge', and 'component split', respectively, and a pair of these to indicate the types of $x$ and $y$. For details on the complete enumeration see [13].

**Time-Sweep Algorithm** To compute all time-varying Reeb graphs, we start with an initial Reeb graph $R_0$ at time $t = 0$, and sweep forward in time modifying the current Reeb graph. With each node in $R_0$ we store information about the segment of the Jacobi curve that contains the corresponding critical point. We think of the sweep forward in time as sliding the Reeb graph forward using the Jacobi curve as a path. In practice, the sweep is achieved by using a priority queue (increasing in time) of birth, death and swap events, at which nodes appear, disappear and changed order respectively. We remove an event from the queue and modify the current Reeb graph to give a new version of the Reeb graph. All Reeb graph versions can be stored using a partially persistent data-structure [10]. Details are in [13].

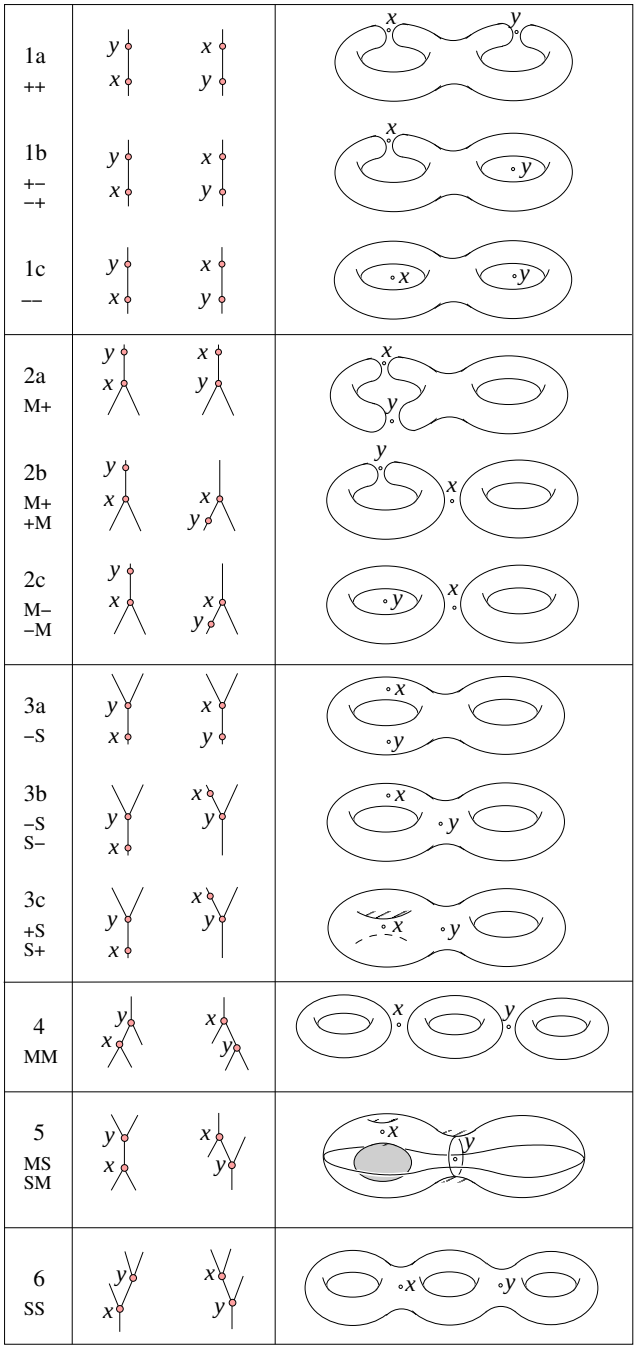


Figure 4: On the left, Reeb graph portions before and after the interchange $x$ and $y$. On the right, level sets at a value just below the function value of $x$ and $y$. In each case, the index of a critical point can be inferred from whether the level set merges (index 1) or splits (index 2) locally at the critical point.

**Betti numbers of level-sets.** Cole-McLaughlin et al.[23] augment the Reeb graph with topological information to enhance visualization. Each arc is labeled with the Betti numbers of the family of level-sets corresponding to that arc. Note that the Betti numbers are computed without actually computing the level sets.

As a part of this dissertation I will investigate methods to extend this idea to time-varying Reeb graphs. Given the Betti numbers for each arc of the initial Reeb graph, we study how the updates to the Reeb graph change the Betti numbers.

# 5 Temporal Simplification Using Time-varying Reeb Graphs.

This section outlines work that needs to be done to perform temporal simplification of time-varying Reeb Graphs. We sketch the basic idea and pose some questions that need to be answered.

The time-varying Reeb graph provides us with a notion of the persistence of a level-set component over time. For an arc $e$, define $\beta(e)$ as its birth-time, and $\delta(e)$ as its death-time. We define the persistence of the arc as $\pi(e) = \delta(e) - \beta(e)$. This definition gives us a mechanism for sorting the arcs in the time-varying Reeb graph in increasing order of persistence and removing "short-lived" arcs based on a threshold persistence. Simplification will play an important role in our ability to visualize large data-sets whose Reeb graphs may also be prohibitively large. By computing progressively simpler Reeb graphs we enable the user to see the "big picture" and then selectively choose certain regions in the volume for a finer visualization.

Further research is needed to investigate the implications of simplifying the time-varying Reeb graph using temporal persistence.

- Is our definition sufficient to capture fleeting level-set components? We expect such fleeting components to be small in size. How do we include a parameter to indicate some measure property, like area or volume, of the level-set component being removed?
- How does deleting an arc affect our ability to find a seed-cell from the Reeb graph?
- Can we extract level-sets that are consistent with the simplified Reeb graph without modifying the data-set? If not, how do we modify the data-set to be consistent with the simplified Reeb graph?

# 6 Expected Results

The main contributions of this dissertation are listed below.

- Classification of changes to the Reeb graph over time. % done - 100
- Algorithm to compute time-varying Reeb graphs. % done - 100
- Implementation of algorithm to compute time-varying Reeb graphs. *% done - 80*
- Extending the concept of path-seeds to time-varying Reeb graphs. *To be done.*
- Computing Betti numbers of the class of level-sets for each edge of the time-varying Reeb graph. % done - 100
- An algorithm to perform temporal simplification on time-varying Reeb graphs. *To be done.*

# 7 Time-varying Volume Data-sets

In this section we consider a few data-sets that we will use to demonstrate capabilities. All data-sets consist of scalar values sampled on a regular grid in space and time. We create a simplicial complex using the sample points as vertices and extend the scalar value to all space and time using barycentric interpolation - i.e the scalar value at any point in the convex hull of the set of points is got by interpolating the scalar value at the vertices of the simplex that contains the point. We consider three classes of data-sets based on their size.

- SMALL: Grids of size $64^4$ and smaller. These include toy data-sets (mainly used to test the algorithm) and sub-sampled versions of larger data-sets.
- MEDIUM: Grids of size $64^4$ to $128^4$. We plan to use these data-sets to stress test our algorithm on desktop computers. Available data-sets: JETDATA - a simulation of a supersonic Jet stream (courtesy of the Advanced Visualization Data Center's data repository), COMBUSTION - simulation of fuel combustion (courtesy of Suresh Menon and Chris Stone at Georgia Tech).
- LARGE: Grids of size $128^4$ and more. We expect to run our algorithm on high performance computer such as those available in Lawrence Livermore National Labs. A data-set in this class is the $1024^4$ sized PPM data-set (courtesy of Valerio Pascucci), which is a simulation of the Richtmyer-Meshkov instability that occurs when a shock wave passes through an interface of two fluids of differing density. The large size of this data-set will certainly require special attention to memory usage, disk access patterns and cache behavior making visualization a challenging task.

# 8 Contributions of Co-Authors

**Time-varying Reeb graphs** The classification of changes to Reeb graph over time was developed jointly with Herbert Edelsbrunner, John Harer, Valerio Pascucci, and Jack Snoeyink. The path-tracing technique to disambiguate node swap cases was developed jointly with Jack Snoeyink, Valerio Pascucci, and Herbert Edelsbrunner.

**Out-of-core streaming isosurface extraction.** This is work submitted to the Second International Symposium on 3D Data Processing, Visualization, and Transmission. The topic is not directly related to the thesis but I mention it here as part of my research in visualizing volumetric datasets. The idea of using the seed-set computation algorithm to generate a disk layout for out-of-core isosurface extraction was developed during discussions with Jack Snoeyink. Martin Isenburg suggested possible disk layout schemes drawing from his experience in compression of hexahedral meshes. Martin also provided help with range coding software used to compress the scalar field. Valerio Pascucci provided valuable input in developing the algorithm.

# 9 Planned Milestones

## 9.1 Progress to Date

- Fall 1999
  - Started work on 6 Degree-of-freedom (6-dof) haptic rendering with Ming Lin.
  - Worked on developing a haptic rendering system for simple polygonal models with Arthur Gregory. Part of the work was later submitted as a paper to IEEE Visualization Conference, 2000
- Spring 2000
  - Developed a system to perform 6-dof haptic visualization of volumetric force-fields.
  - Worked with Stephen Ehmann on developing a 6-dof interaction system in a dynamic environment.
  - The paper describing this work was submitted to IEEE Visualization
- Summer 2000
  - Worked on a paper submission to "Touch in Virtual Environments", a workshop on haptics held at the Integrated Media System Center, University of Southern California.
  - Investigated building bottom-up bounding volume hierarchies.
- Fall 2000
  - Started work with Jack Snoeyink on building pentatope meshes for time-varying volumetric datasets using important point insertion.
- Spring 2001
  - Worked on computing seed-sets for fast isosurface extraction in time-varying volumetric datasets.
  - Presented 6-dof haptic work at "Touch in Virtual Environments", a workshop on haptics held at the Integrated Media System Center, University of Southern California.
- Summer 2001
  - Worked on predicting potential ligand binding conformations in the humane Pregnane X Receptor (hPXR) molecule with Robert-Paul Berretty, and David Hsu.
- Fall 2001
  - Satisfied Integrative Paper (IP) requirement.
  - Worked on using Graphics hardware to compute seed-sets for fast isosurface extraction.
- Spring 2002
  - Continued work on using Graphics hardware to compute seed-sets for fast isosurface extraction.
- Summer 2002
  - Summer internship at Lawrence Livermore National Laboratory (LLNL). Developed code to compute Jacobi set of time-varying volumetric datasets
- Fall 2002
  - Started study of time-varying Reeb graphs.
  - Worked on integrating Sphere-based Delaunay construction in 4D with mesh refinement to build pentatope meshes for time-varying volumetric datasets.
- Spring 2003

  - Experiments on building pentatope meshes for time-varying volumetric datasets.
  - Developed algorithm to compute seed-sets for time-varying volumetric data-sets defined on a regular grid.
  - Wrote a manuscript describing the seed-set algorithm and results. This is to be completed and published. (by end of April 2004)
- Summer 2003
  - Completed teaching requirement.
  - Summer internship at Lawrence Livermore National Labs
  - Worked on developing theory and algorithm to compute time-varying Reeb graphs. A paper describing the theory and algorithm was written, and is to be published. (Aiming for submission to Visualization 2004.)
  - Developed code to create a disk layout for volumetric data-sets to enable out-of-core isosurface extraction. This work is to be continued and published.
- Fall 2003
  - Implementation of time-varying Contour tree algorithm. (80% done)
  - Submitted paper on time-varying Contour trees to Symposium on Computational Geometry. (Dec 12th 2003) [13]
  - Worked on out-of-core isosurface extraction.
- Spring 2004
  - Submitted paper on organizing volumetric scalar data for out-of-core streaming isosurface extraction to Second International Symposium on 3D Data Processing, Visualization, and Transmission.

## 9.2 Future Research Plans

- Summer 2004
  - Ph.D. Proposal
  - Attend Workshop in Banff, Canada.
  - Present paper at ScG in June, 2004.
  - Complete implementation of time-varying Contour tree algorithm.
  - Work on computing Betti numbers of the class of level-sets for each edge of the time-varying Reeb graph
  - Extending the concept of path-seeds to time-varying Reeb graphs.
  - Work on temporal simplification of time-varying Reeb graphs.
  - Publish results.
  - Write Dissertation.
- Fall 2004
  - Ph.D Oral Exam.
  - Write Dissertation.
  - Defend (December 2004).

# References


[1] P. S. ALEXANDROV. "Combinatorial Topology.", Dover, Mineola, New York, 1998.

[2] C. L. BAJAJ, V. PASCUCCI, AND D. R. SCHIKORE. Fast isocontouring for improved interactivity. In *Proc. of the 1996 Sympos. for Volume Vis.*, pages 39–46, Oct 1996.

[3] C. L. BAJAJ, V. PASCUCCI, AND D. R. SCHIKORE. The Contour Spectrum. In *Proc. of IEEE Vis.*, pp. 167–175, 1997.

[4] C. L. BAJAJ, V. PASCUCCI, AND D. R. SCHIKORE. Seed sets and search structures for optimal isocontour extraction. Technical report, University of Texas, Austin, 1999. http://www.ticam.utexas.edu/CCV/papers/cont-tog.pdf.

[5] C. L. BAJAJ, A. SHAMIR, AND S. BONG-SOO. Progressive tracking of isosurfaces in time-varying scalar fields. Technical report, University of Texas, Austin, 2002. http://www.ticam.utexas.edu/CCV/papers/Bongbong-Vis02.pdf.

[6] H. CARR, J. SNOEYINK AND U. AXEN. Computing contour trees in all dimensions. *Comput. Geom.* **24(2)** (2003), 75–94.

[7] H. CARR, AND J. SNOEYINK. "Path seeds and Flexible Isosurfaces: Using Topology for Exploratory Vis." In *Proc. of Eurographics Vis. Sympos. 2003*, 2003.

[8] K. COLE-MCLAUGHLIN, H. EDELSBRUNNER, J. HARER, V. NATARAJAN, AND V. PASCUCCI. "Loops in Reeb graphs of 2-manifolds" In *Proc. of the 14th Annual ACM Sympos. on Computational Geometry*, pages 344-350, 2003

[9] T. H. CORMEN, C. E. LEISERSON, AND R. L. RIVEST. *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts, 1994.

[10] J. R. DRISCOLL, N. SARNAK, D. D. SLEATOR, AND R. E. TARJAN. "Making Data Structures Persistent", in *Journal of Computer and System Sciences* **38**, pp. 86–124, 1989.

[11] H. EDELSBRUNNER, AND E. P. MÜCKE. "Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms", in *ACM Trans. Graphics* **9**, pp. 66–104, 1990.

[12] H. EDELSBRUNNER AND J. HARER. Jacobi sets of multiple Morse functions. In *Foundations of Computational Mathematics*, ed. F. Cucker, Cambridge Univ. Press, England, to appear.

[13] H. EDELSBRUNNER, J. HARER, A. MASCARENHAS, AND V. PASCUCCI "Time-varying Contour Trees for Continuous Space-time Data." In *The 20th ACM Sympos. on Computational Geometry*, *to appear*.

[14] R. S. GALLAGHER. Span filtering: An efficient scheme for volume visualization of large finite element models. In G. M. Neilson and L. Rosenblum, editors, *Vis. '91 Proc.*, pages 68–75, Oct 1991.

[15] M. GILES AND R. HAINES. Advanced interactive visualization for cfd. In *Computing Systems in Engineering 1*, volume 1, pages 51–62, 1990.

[16] C. T. HOWIE AND E. H. BLACK. The mesh propagation algorithm for isosurface construction. In *Computer Graphics Forum 13, Eurographics '94 Conf. Issue*, pages 65–74, 1994.

[17] T. ITOH AND K. KOYAMADA. Automatic isosurface propagation using an extrema graph and sorted boundary cell lists. In *IEEE Transactions on Vis. and Computer Graphics 1*, volume 1, pages 319–327, Dec 1995.

[18] W. E. LORESEN AND H. E. CLINE. "Marching cubes: A high resolution 3d surface construction algorithm." In M. C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proc.)*, volume 21, pages 163–169, July 1987.

[19] Y. LIVNAT, H. W. SHEN, AND C. R. JOHNSON. A near optimal isosurface extraction algorithm for unstructured grids. In *IEEE Transactions on Vis. and Computer Graphics*, volume 2, pages 73–84, 1996.

[20] Y. MATSUMOTO. "An Introduction to Morse Theory.", Translated from Japanese by K. Hudson and M. Saito, Amer. Math. Soc., 2002.

[21] J. MILNOR. "Morse Theory." Princeton Univ. Press, New Jersey, 1963.

[22] J. R. MUNKRES, "Elements of Algebraic Topology", Addison-Wesley, Redwood City, California, 1984.

[23] V. PASCUCCI AND K. COLE-MCLAUGHLIN. "Efficient computation of the topology of level sets" In *Algorithmica* to appear.

[24] G. REEB. Sur les points singuliers d'une forme de Pfaff complèment intégrable ou d'une fonction numérique. *Comptes Rendus de L'Académie ses Séances, Paris* **222** (1946), 847–849.

[25] H. W. SHEN. Iso-surface extraction in time-varying fields using a temporal hierarchical index tree. In *IEEE Proc. of Vis. '98*, pages 159–166, Oct 1998.

[26] P. SUTTON AND C. HANSEN. Isosurface extraction in time-varying fields using a temporal branch-on-need tree(t-bon). In *IEEE Proc. of Vis. '99*, pages 147–153, 1999.

[27] S. P. TARASOV AND M. N. VYALYI. "Construction of Contour Trees in 3D in $O(nlog(n))$ steps" In *Proc. of the 14th Annual ACM Sympos. on Computational Geometry*, pages 68–75, 1998.

[28] M. VAN KREVELD, R. VON OOSTRUM, C. L. BAJAJ, V. PASCUCCI, AND D. R. SCHIKORE. Contour trees and small seed sets for isosurface traversal. In *The 13th ACM Sympos. on Computational Geometry*, pages 212–220, 1997.

[29] C. WEIGLE AND D. BANKS. Extracting iso-valued features in 4-dimensional scalar fields. In *IEEE Vis. Conf.*, pages 51–62, 1995.

[30] J. WILHELMS AND V. GELDER. Octrees for faster isosurface generation. In *ACM Transactions on Graphics*, volume 11, pages 201–227, 1992.