Topological Visualisation Techniques for Volume Multifield Data

Dean P.Thomas [1][2]
798295@swansea.ac.uk

Robert S.Laramee [1]
r.s.laramee@swansea.ac.uk

Rita Borgo [3]
rita.borgo@kcl.ac.uk

Simon J.Hands [2]
s.j.hands@swansea.ac.uk

[1] Department of Computer Science, Swansea University, Swansea, United Kingdom

[2] Department of Physics, Swansea University, Swansea, United Kingdom

[3] Department of Informatics, Kings College London, London, United Kingdom

Abstract

This survey paper provides an overview of topological visualisation techniques for scalar data sets. A review of the topological ambiguities in Marching Cubes algorithms forms a basis for introducing topology-led approaches to visualisation. Topological algorithms are used to reduce scalar fields to a skeleton by mapping critical changes in the topology to the vertices of graph structures. These can be visualised using graph drawing techniques or used as a method of seeding meshes of distinct objects existing in the data. Many techniques are discussed in detail, beginning with a review of algorithms working on scalar fields defined with a single variable, and then generalised to multivariate and temporal data. The survey is completed with a discussion of methods of presenting data in higher dimensions.

Keywords: indirect volume rendering, topology driven visualisation, multivariate visualisation, contour tree, Reeb graph, Reeb space, joint contour net, Reeb skeleton, scalar data

1. Introduction

The Marching Cubes (MC) algorithm [1] is a long established method used in indirect visualisation for creating mathematical models of data existing in a scalar field. Early research centred upon improvements for the presentation of data from medical sources such as CT and MRI scans. Whilst well suited for approximating isosurfaces on smooth functions, MC derived algorithms typically struggle to accurately capture features such as sharp edges and corners. These features can often to lead to poorly shaped triangles. Various approaches have been suggested for correcting these limitations including *Extended Marching Cubes* [2], *Dual Marching Cubes* [3], and *Cube Merging* [4]. The algorithm continues to be refined and improved for various purposes, and remains an active field of study [5].

Besides improvements to model quality, a recurring theme in research linked to the algorithm is identifying and solving topological ambiguities in the models it creates.

Typical ambiguities result in the absence of triangles between two surfaces existing in opposing corners of a MC cell. This can lead to creation of a model which contains two separate objects, when in reality the model should be a single joined object. Many approaches have been suggested for overcoming these ambiguities including the use of Marching Tetrahedra [6]. The use of topological algorithms removes the possibility for ambiguities, whilst providing a method for seeding disjoint contours using similar lookup tables to those of Marching Cubes. Algorithms for surface generation, such as Marching Cubes, can also be generalised for use in multivariate data sets.

The remainder of this chapter is structured as follows. Section 2 highlights the main issues with creating topologically correct surfaces using MC algorithms. Section 3 describes topological visualisation techniques for scalar fields that solve the ambiguity problems of MC. In Section 4 we introduce Fibre Surfaces, a multivariate extension to conventional MC, which is used as a basis for extending topological techniques to cover multivariate data sets in Section 5. Section 6 considers displaying the output of topological algorithms from higher dimensional data sets. The chapter is concluded in Section 7.

2. Topological Irregularities in Marching Cubes

One particularly problematic aspect of the Marching Cubes algorithm is that it is prone to creating isosurfaces with holes. This was a point first highlighted in a letter [7] to the Computer Graphics journal shortly after the algorithm was published. A later modification by Nielson and Hamman [8] proposed to address the issue using a technique named the *Asymptotic Decider*. The MC algorithm does not cope well with triangulation methods in ambiguous configurations of the lookup table. Of the original fifteen configurations present within the MC lookup table, five configurations are prone to generating ambiguous faces. Further investigation of the problem revealed that in total there were an additional twenty-six variations on those already present. The ambiguities can typically be addressed by inserting further branches into the lookup table without a noticeable increase in running time.

Chernyaev's [9] approach to alleviating topological inconsistencies in marching cubes was *Marching Cubes 33 (MC33)*, named so because the modification provided a lookup table of thirty-three configurations. Fixing internal ambiguities was the main focus of this algorithm and it was largely similar in implementation to the asymptotic decider. An example of an internal ambiguity is the case where opposing vertices are marked as in / out of the surface but it is not possible to tell if they are connected using internal triangles. Using a bilinear interpolation across a plane parallel to a face Chernyaev suggested a possible solution to the problem, describing the corrected ambiguous configurations.

Lewiner et al. [10] showed that the MC33 algorithm was unable to create topologically correct surfaces in all situations. Their algorithm extended MC33 by testing for additional internal ambiguous situations, whilst optimising the earlier algorithm by removing redundant tests. The resultant algorithm claimed to be topologically correct by removing cracks and topological inconsistencies. Further topological irregularities were highlighted in the MC33 algorithm by Etiene et al. [11] with rectifications proposed in an algorithm known as *Corrected-MC33 (C-MC33)* by Custodio et al. [12]. The corrected algorithm solved ambiguities within configurations 10, 12, and 13.

Alternative approaches to MC33 have been proposed to correct topological irregularities using additional triangles within configurations with ambiguities. Whilst the addition of adding polygons to the mesh impacted on raw processing speed, Montani et al. [13] felt that the added simplicity of avoiding branching in ambiguous configurations counteracted this. In practice when compared to reference functions and using real datasets, little computational overhead became evident in comparison to the original MC algorithm. An extension to both the original MC algorithm and the asymptotic decider was later released by Nielson [14], that reverted back to a branching method. This further solved the ambiguities that were liable to cause holes in isosurfaces by extending the linear interpolation of values, and later bilinear interpolation of faces, with a trilinear interpolation within the interior of voxels. Using several test cases for each of the look-up table configurations Nielson was able to prove that topologically sound surfaces could be generated using this method. This was later confirmed as being topologically correct by Carr [15], who described the method for completing the trilinear interpolation process as dividing the existing cubes into smaller blocks with simpler topology. However, the trilinear interpolation method showed differences in outputs to those proposed by Montani et al. [13].

3. Topology driven visualisation

Algorithms for computing isosurfaces, otherwise known as level sets, are prone to topological inaccuracies, as demonstrated in Section 2. During the construction of an isosurface no topological information is used; hence, an isosurface is unable to distinguish individual connected regions. Morse theory provides a method for computing the topology of scalar data, as we sweep through the isovalue range, by considering the gradient and second derivatives of the level set at each vertex in the scalar field. Through the use of tetrahedral mesh cells (for 3D scalar fields) we can approximate the isosurface as a piecewise linear function between sampling points. However, at the boundary between cells this approach invalidates a key condition of Morse theory which requires a function to have derivatives defined at all sampling points. Work by Edelsbrunner et al. [16] and Bremer et al. [17] provides a mechanism for handling these limitations, allowing us to consider the data as a Morse function.

Topology driven visualisation is used to capture characteristics of a data set using basic topological invariants such as the number of holes or points in a connected region. This information provides a means for obtaining a skeleton of the data which can then be used to construct visualisations in various forms. Use of topological visualisation techniques can see a reduction in the size of a large data set, as only key features need to be retained. By using topological structures to represent a data set intuitive methods can be used for speeding up computations; for example, by allowing many parts of a data set to be rendered in parallel. A further increase in efficiency is provided by the ability to bypass redundant computations — such as the processing of empty cells in MC. Increases in rendering performance come with the overhead of the need to do a one-time pre-processing of the data.

Many topology based algorithms rely on abstract graph based structures for storage of data. In many situations the graph structures can be viewed directly to form an overview of the data in an easily perceived format. For those with little prior knowledge of topological visualisation techniques it can be hard to form an initial understanding of the data using such formats. In order to better understand the data it is common to directly link the graph visualisation, either statically or dynamically, with a rendered view of the data. However, difficulty in understanding topological data representations can be further complicated when a data set is large in size. Hence, other approaches are sometimes used that try to take advantage of human perception, such as topological landscapes.

3.1 Terminology

In order to present the algorithms and structures used in topological visualisation the following key terminology is used.

Simplex The simplex is a generalisation of the triangle or tetrahedron to n dimensions. Properties of simplexes in dimensions 0 (a point) through to 4 (a Pentatope) are given in Table 1. Computational topology often refers to meshes as simplicial complexes — a set of n simplexes glued together at their boundaries.

Betti numbers Enable us to categorise the topology of a mesh existing in n dimensions by examining the connectivity of its structure as a number of n dimensional simplexes. Each dimension, from 0 to $n-1$, has its own Betti number β_n associated with it to represent the quantity of cuts required through a mesh in order to separate it into two parts. We can consider these numbers as representing the number of holes in the mesh in n dimensions (Table 2). Most topological visualisation algorithms consider only the 0^{th} dimensional Betti number, used to represent connectivity. However, it is possible to augment the graphs with higher order Betti numbers to provide feedback on changes in the mesh such as the morphing from a sphere to a torus (Table 3). In this work we are primarily only interested in the zeroth dimensional Betti number β_0.

Genus This is a way of expressing the number of holes in a surface. In three dimensions, a sphere is genus 0, and torus genus 1. Surfaces with additional holes are commonly referred to as an n-torus.

Critical point The sampling points in a scalar field $f(x)$, at which the function value is defined, can be categorised into two types. Critical points have a local gradient $f'(x) = 0$ and represent positions where the topology of the level set changes. These relate to local extrema or saddle points in the data; the second derivative $f''(x)$ allows us to classify if the critical point is a local minima $f''(x) > 0$ or a local maxima $f''(x) < 0$. In the case where a critical point is a local minima all neighbouring vertices will have a higher function value, whilst a local maxima means that all surrounding vertices are lower in value. Regular points have a non-zero gradient and have no overall effect on the connectivity the level set.

Contour The boundary of connected regions of a level set, also known as the 0-dimensional homology group, are known as contours. Each individual contour is an element in the level set that represents a distinct topological object within the isosurface.

Topological persistence The most simplistic way of defining topological persistence is as a quantitative measure of importance of each contour. In literature topological persistence is often known as geometric measures [18] due to the fact it relates to measures such as volume or surface area. It is possible to compute persistence values directly from the meshes representing each contour by performing a piecewise sum of all cells in a connected region. Alternatively, the persistence can be approximated by counting the number of regular vertices making up a connected region. Persistence measures are often used as a method of eliminating noise from the topological structure by iteratively removing the contour with smallest persistence value.

Name	Dimension n	Common Name	Vertices	Edges	Faces	Cells
0-simplex	0	Point	1	-	-	-
1-simplex	1	Line-segment	2	1	-	-
2-simplex	2	Triangle	3	3	1	-
3-simplex	3	Tetrahedron	4	6	4	1
4-simplex	4	Pentatope (or 5-cell)	5	10	10	5

Table 1 Classification of simplexes in dimensions zero to four.

Betti Number	Associated Simplex	Property
β_0	Point	Connected components
β_1	Line-segment	Holes
β_2	Triangles	Voids

Table 2 Betti numbers β_0, β_1, and β_2 and associated concepts of connectivity.

Topological concept	β_0	β_1	β_2
Sphere	1	0	1
Torus	1	2	1

Table 3 Betti numbers used to describe the sphere and torus.

Name	Dimension	Common name
0-manifold	0	Point
1-manifold	1	Polyline
2-manifold	2	Polygon
3-manifold	3	Polyhedron
4-manifold	4	4-Polytope

Table 4 Classification of manifolds in dimensions zero to four.

Most of the techniques and structures discussed in this section can be generalised for use in any number of dimensions — a list of general terms is provided in Table 4.

3.2 Merge Trees

One of the fundamental data structures in computational topology are merge trees, commonly used to describe two similar concepts. Merge trees track connected level sets within the data as the isovalue is swept across the scalar range. The *join tree* captures connected sub-level sets, or regions, of the data below a specific isovalue threshold. Similarly the *split tree* captures connected super-level sets.

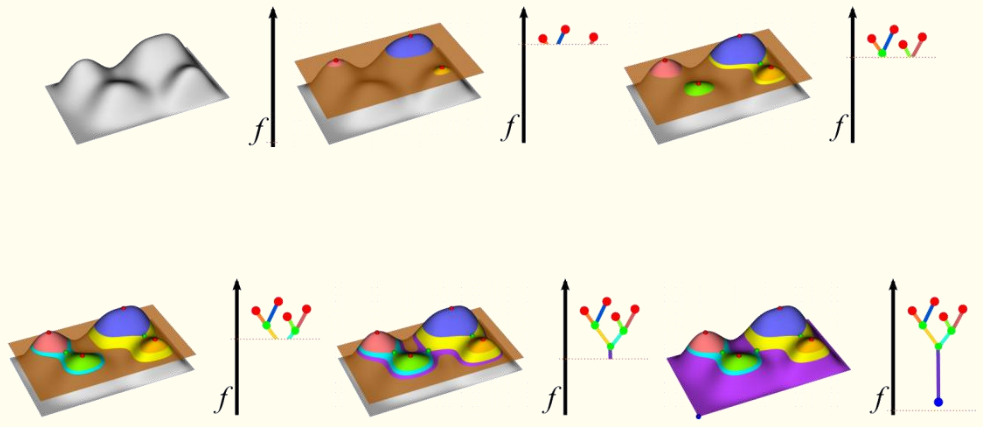

Figure 1 A visual representation of the merge tree algorithm for the function shown in the top-left. The merge tree captures each of the local maxima as the function height is reduced (moving left-to-right, top-to-bottom). A real world analogy is to think of the isovalue as the water level in a valley that gradually drains to reveal the peaks of multiple hills. Image courtesy of Landge et al. [19].

Merge trees present a simple method for capturing topological information as the arcs in a tree-like structure. This can then be used to compute zero-dimensional persistence measures representing the number of connected components in a region [20]. Merge trees are often used as a computation step in the construction of more complex topological data structures such as the contour tree, described in detail in Section 3.3. Alternatively, merge trees can be used directly for analysis and feature extraction in complex data sets [19].

A visual metaphor for understanding the concept of merge trees is given in Figure 1. To capture the merge tree, in this case the join tree of the grey function (top-left), the isovalue is gradually decreased from the global maxima. In the top-centre image this has revealed three local maxima, each denoted as a red critical point. In the top right image a fourth local maxima has been revealed (the green contour) and the red and blue structures have merged, this is marked by a node in the graph. In the bottom left image, the pink and green surfaces have merged to form the cyan region. The bottom centre image sees the small intervals represented by the cyan and yellow surfaces merged into one large purple region. If the isovalue continues to be reduced it is possible to observe no further changes in the topology until reaching the global minima represented by the blue graph node.

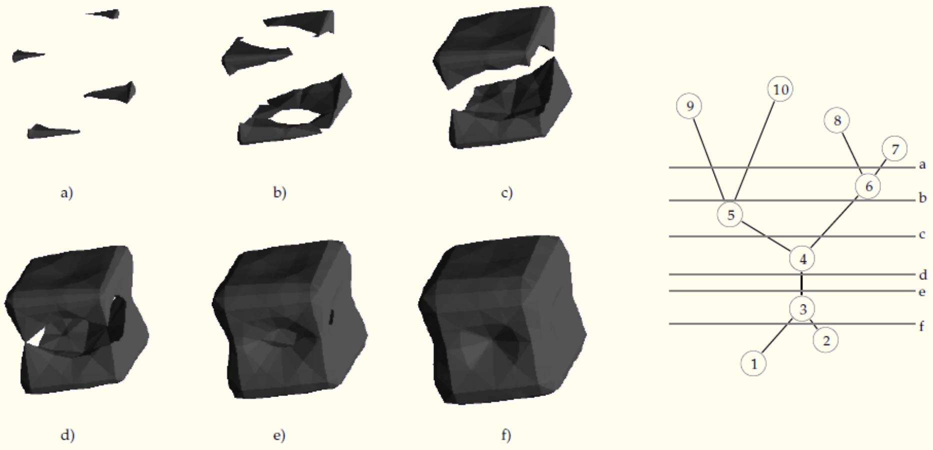

Figure 2 Left: a surface that varies as the function value is varied. Right: a labelled contour tree representation of the surface. Image courtesy of Carr et al. [21]**.**

3.3 The contour tree

The contour tree, introduced by Boyell and Ruston [22] uses concepts from Morse theory [23], [24] to capture changes in the topology of scalar field at critical points in the data. The tree structure allows the tracking of splits and joins in the topology as the isovalue is varied by tracing a path through the graph. Figure 2 demonstrates how individual contours in the data are captured by the contour tree using a one-to-one correspondence with edges. Each leaf in the contour tree represents an individual local extrema and critical vertices represent a change in the connectivity of the level set. Typically, the contour tree does not contain information regarding changes in topological genus; however, this information can be added to the output [25].

We can formally describe the structure of the contour tree as follows:

- Vertices or *supernodes*
    - Leaf nodes correspond to:
        - Local maxima where a contour is created.
        - Local minima where a contour is destroyed.
    - Interior nodes correspond to:
        - A saddle point where one more contours are created as a contour splits into two or more disjoint contours.
        - A saddle point where one or more contours are destroyed as two or more disjoint contours merge into one.
- Edges or *superarcs*
    - Represent a single contour between the supernode where it is created and the supernode where it is destroyed.

In addition to optimizations of contour tree computation, research is focused on approaches for storing the contour tree hierarchy, allowing it to be traversed in an efficient manner. A number of related approaches have been proposed including Kd-trees, segment trees, and interval trees. These formats prove problematic as they can require a large amount of storage and are difficult to navigate. An alternative method is proposed in [26] using the observation that an isosurface is the level set of a continuous function in 3D space; therefore, a whole contour can be traced from a single element. Vertices where contours can be built from are given the name seeds; algorithms for computing seeds relate to work in image processing requiring similar storage facilities. An entire contour can be computed for the specified desired isovalue from the seed that is bounded by the two supernodes representing the critical points of the contour. Results showed that by using this method the number of seeds required for representation were in most cases significantly reduced. However, often the overall storage size of the contour tree was increased in comparison to other methods.

Carr et al. [21] investigated the use of contour trees in higher dimensional data sets, whilst also improving upon the algorithm proposed in [27]. This also introduced the concept of *augmented contour trees*, an extension that added non-branching vertices at non-critical points in the data to provide additional values for isosurfaces to be seeded from. This feature was built into a GUI allowing the user to identify regions of interest, using colour coding and to distinguish them as directed by the user [28]. Carr and Snoeyink [29] use the contour tree as the underlying data structure to generate object meshes using path seeds. The use of the contour tree can be extended to finding paths between two points given condition clauses, such as a minimum or maximum values, on a function defining a landscape [30].

Chiang et al. [31] introduce a modified contour tree construction algorithm that improves processing time by only sorting *critical vertices* in the merge tree construction stage. This vastly increases processing speeds in very large data sets. However, as observed by the authors, critical vertices are difficult to identify in data with dimensionality of four or more.

Further increases in speed are offered by storing multiple seeds for the monotone path construction algorithm [32] used to generate surfaces. An increased storage overhead is required; however, a surface does not require complete re-extraction each time the isovalue is changed. Recently Gueunet et al. [33] presented a multithreaded approach to computing the augmented contour tree using a shared memory approach. Initial evaluations of the "contour forest" algorithm showed quicker computations achievable using the approach; however, at present the extent of the speed up is limited by load imbalance and redundant computations.

For a given technique of subdividing the input domain, the output of the contour tree algorithm is fixed. However, in the case that a different technique is used to define the neighbourhood of sampling points the location of the critical point may change trivially. For example, changing the method of defining neighbours during split / merge tree computations can slightly alter the location of critical points. The underlying structure of the tree, the number of supernodes and superarcs, remains fixed for a given input with only the geometric locations of supernodes changing.

3.4 The Reeb graph

Using the contour tree comes with the limitation that the input must be defined on a *simply connected domain* that is free of loops. Limitations in the merge tree computation phase of the contour tree algorithm prevent it from correctly handling data without a boundary. However, the Reeb graph [34], a generalisation of the contour tree, can be used to compute the topology of scalar data in such situations [35]. As with the contour tree, the Reeb graph can be applied to models of any dimension provided it is represented on a simplicial mesh [36]. The Reeb graph has also been used to assist in the design of transfer functions in volume visualisation [37], by assigning opacity based upon how many objects were obscured by nested surfaces and their proximity to the edge of a scalar field.

The first use of the Reeb graph for encoding topological features for visualisation purposes was by Shinagawa et al. [38] where the structure was used as a way of representing objects obtained from computerized-tomography (CT) sources [39]. An online algorithm is given by Pascucci et al. [36] that allowed the Reeb graph of a function to be computed using a streaming approach with the output continually updated as additional data points are added. Efficiency of the algorithm is optimal for two dimensional inputs and the streaming nature of the algorithm limits peak memory usage. However, the $O(n^2)$ complexity of the algorithm, where n is the number of triangles, means it performs less favourably for higher dimensional inputs.

A key feature of many Reeb graph algorithms is that a *2-skeleton* of the input volume is used to define the relationship between vertices, edges and triangles. The same restriction applies to scalar fields in n dimensions; hence, provided a triangulation of the data is available the algorithm will be able to compute a correct n dimensional Reeb graph. Reeb graph algorithms can largely be split into two groups; those that are *sweep based*, requiring the maintenance of level sets, and those that use a *split and compute* method to collapse the problem to that of a *contour tree* computation. A third alternative was given by Harvey et al. [40] that randomly collapsed triangles for arbitrary 2-skeletons to improve the running time to $O(m \log m)$.

The sensitivity of Reeb graph computations to the topological genus of the input domain was first demonstrated by McLaughlin [35] in the case of a non-simply connected domain, such as that representing a sphere or torus. Additionally, the number of loops present in the data as a result of the Morse function can further extend the complexity of the algorithm. In order to address this undesired effect the Reeb graph computation can be reduced to that of the simpler contour tree algorithm. This approach was first used by Tierny et al. [41] where they performed "loop-surgery" by making symbolic cuts during the *merge tree* step of the algorithm. The symbolic cuts could then be stitched back together to retrieve a topologically correct Reeb graph of the data. An alternative approach, capable of generalising to higher dimensional inputs, was later proposed by Doraiswamy and Natarajan [42] that explicitly maintained level sets, eliminating the need for a loop-surgery pre-processing step.

More recently Doraiswamy and Natarajan [43] offered an improved algorithm for constructing the Reeb graph of n-dimensional scalar fields, reverting to the split and compute contour tree technique, as originally proposed in [41]. A further optimised algorithm [44], using a variation of the split and compute technique, used the join tree of the data to identify potential loops. An important modification upon the technique developed by Tierny et al. [41] was the segmentation of domain into multiple loop free contour trees, instead of a single contour tree, thus enabling multiple regions of the input to be computed in parallel.

Output from Reeb graph algorithms remains fixed provided the simplicial subdivision defined upon the input domain does not change. As is the case with the contour tree, variations can slightly alter the location of the critical vertices without changing the overall structure of the graph.

3.5 Seeded contours

Contours, as opposed to isosurfaces, can be grown using values extracted from the scalar field to produce topologically correct meshes. De Berg et al. [30] made the observation that a whole contour could be traced starting from a single *seed* element. Wyvill et al. [45] generate contour surfaces as a two-stage process; first, the core region is flood filled by evaluating neighbouring cells for inclusion in the level set. In the second stage boundaries are calculated using look-up tables similar in form to those of the MC algorithm [1].

Contour trees are a reliable source of seed cells for propagation algorithms similar to those of Wyvill et al. [45]; hence, the contour tree can be used to generate surfaces for each superarc at a given isovalue. Carr and Snoeyink [29] use the contour tree as the underlying data structure to store and generate object meshes. Rather than using *minimal seed sets* as used by van Kreveld et al. [26] to generate contours, an alternative method was deployed using *path seeds*. Carr et al. [21] also investigated the use of contour trees in higher dimensional datasets, whilst also improving upon the algorithm proposed by Tarasov and Vyalyi [27]. This enables users to identify individual contours of interest and distinguish them accordingly [28].

The *flexible isosurface* [46] is a technique for combining many of the features discussed in this section with the concepts of *topological persistence* (see Section 3.6) into a single interface. Simplification methods, as discussed in [18], can also be applied to the data so as to display isosurfaces in a meaningful way. This approach allows each contour to be hidden or displayed according to the user preferences at runtime. In order to further aid data exploration contours can also remain fixed in view as the global isovalue is varied. Colour can be applied to each surface (or a sub-tree of the main contour tree) to allow assignment of meaning to contours by the user by providing a simple grouping mechanism.

3.6 Topological persistence and simplification

Different scientific fields of study often define the interesting features and attributes in a domain specific form. However, some features are invariant and can be useful in any field of science. The contour spectrum [47] was introduced as a method of relaying quantitative information about individual contours in scalar data including surface area and volume. Carr et al. [48] directly compare isosurface statistics against raw histograms of scalar data for a number of data sets. Measurements evaluated included the cell intersection count, triangle count and isosurface area. It was found that using these measures a truer distribution of the scalar field could be computed. An improvement was given by Meyer et al. [49] using concepts from geometric measure theory that minimised the effect of noise on the observed distributions. The key to this improvement was introducing a normalisation of the individual contour statistics to the domain average.

Scalar field data often contains noise; when partitioned using a topology sensitive algorithm this manifests in the generation of a large quantity of small objects present at a limited range of isovalues. Methods for reducing noise were first proposed by Edelsbrunner et al. [20] using an iterative process that performed topological simplification by assessing the Betti-numbers of topological objects. The goal of the algorithm is to simplify shape whilst preserving the underlying topological features of data existing on a triangular mesh.

Carr et al. [18] proposed the use of concepts originally discussed as part of the contour spectrum [47], such as enclosed volume and surface area, as an aid for noise removal. Objects are queued in ascending order, according to the user selected measure, and iteratively removed until a terminating level of simplification is achieved. The effects of three different simplification features are compared using X-ray data from a human skull, where it was found that use of the isovalue range persistence measure (Figure 3) could result in the removal of important features such as the eyes. Carr et al. remark that simplification techniques are sensitive to the source and should be chosen using domain specific knowledge. The technique is suited to n dimensional data; however, time variate data requires additional considerations in the simplification process due to connectivity between time steps.

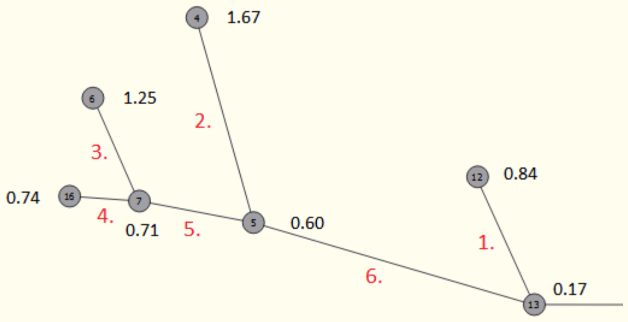

Figure 3 An example of a topological persistence measure defined on the superarcs of a contour tree or Reeb graph. The measure of persistence defined here uses the isovalue range - this is given as the difference of isovalues of the two critical vertices defining the superarc (see Table 5).

Superarc		Top supernode		Bottom supernode	
id	persistence	id	isovalue	id	Isovalue
1	0.67	12	0.84	13	0.17
2	1.07	4	1.67	5	0.60
3	0.54	6	1.25	7	0.71
4	0.03	16	0.74	7	0.71
5	0.11	7	0.71	5	0.60
6	0.43	5	0.60	13	0.17

Table 5 Persistence measures associated with Figure 3.

3.7 Topology in Direct Volume Rendering

The techniques discussed in Section 3.5 model topological objects as meshes, meaning they are well suited to indirect volume visualisation. However, topology based approaches can also be applied to data displayed using direct volume rendering approaches.

A segmentation algorithm, such as the contour tree or *volume skeleton tree*, is first used to look for boundaries between objects in the scalar field topology. Following this, traditional direct volume rendering techniques can be applied to the data based upon attributes of the identified objects. Takeshima et al. [50] use attributes such as the number of equal valued contours and occlusion to assign levels of opacity to objects. The net effect was that the outermost objects were assigned lower opacities so as to not obscure features centred in the volume. A more flexible approach was used by Weber et al. [51] applying distinct transfer functions to each object, or topological zone, as directed by the user. This customization, implemented via the user interface, enables grouping of similar features and related components using colour and transparency.

3.8 Temporal univariate scalar data

Recent interest in topological visualisation research has focused upon the comparison of scalar data through topological methods to assign a degree of similarity to data. This has potential uses in different scientific domains including physics, chemistry, and climate science, which often feature a temporal component. The merge tree, with its simple data-structure, presents an attractive method for computing topological differences between data sets. Beketayev et al. [52] define a distance measure between merge trees with potential applications in a range of scientific disciplines. The algorithm uses a *branch decomposition* approach to deconstruct the merge tree into multiple sub-graphs, each a descending path from a saddle to a leaf vertex. Each branch decomposition can then be scored via an adapted form of the edit distance [53] between two graphs. A recursive algorithm then tests if two merge trees are within an epsilon value to determine if they are classified as similar.

This approach was further extended by Saikia et al. [54] to produce the *extended branch decomposition graph*, a union of all individual sub-trees. This data structure allows quicker comparison between merge trees by computing multiple similarity thresholds in parallel. In addition the extended branch decomposition graph improves upon memory usage by reducing the redundancy found in multiple disjoint branch decompositions. Branch decomposition methods have also be applied to the more complex contour tree to compute similarity and symmetry in scalar topology [55].

The method used by Beketayev et al. [52] had a runtime of $O(n^5)$, where a merge tree contains n nodes, this was lowered to $O((n \log n)^2)$ in related work [54]. Potential downfalls to the optimised algorithm are that instabilities can arise from permuted forms of branch decompositions, this is handled in [52] by considering all possible permutations at the cost of scalability. The optimised algorithm uses the *extended branch decomposition graph*, a union of all possible branch decomposition graphs, to store all branch decompositions in a single tree structure. Two potential uses for the distance measure are suggested; identification of similar structures within a single data set or, for time variate data, repetition between time steps.

Fluid dynamics is an area of science that can benefit from visualisation of 3D volumes with a time component. The complexity of mixing two fluids is one specific problem that can be better understood using topological methods as the system evolves over time [56]. Topological analysis makes it possible to take a slice of the volume at a given time-step and count the number of bubbles present. Alternatively, topological analysis enables tracing of properties, such the volume and centre of gravity of individual bubbles, throughout the simulation.

4. Marching Cubes methods for multivariate data

Typical MC algorithms are applicable to scalar fields where the data at each point is a single variable. However, it is also possible to visualise fields where each point is associated with multiple variables. For this purpose the concept of the isosurface generalises to that of the *fibre surface* in multi-field data sets. Fibre surfaces utilise indirect volume techniques to create geometric models of the multi-field, in addition to generating visual representations this allows geometric properties of the multi-field to be queried.

A MC based algorithm was presented by Carr et al. [57] that allowed the capturing of fibres for bi-variate data sets. The captured fibre surfaces are geometrical in nature, rather than topological; hence, they don't take connectivity into account.

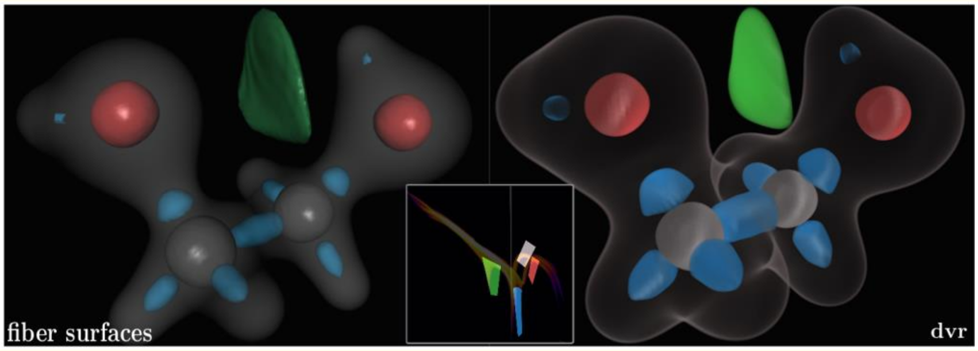

Figure 4 A direct comparison of a fibre surface and a DVR render of an ethane-diol molecule simulation. The central image shows a scatterplot of the data. Image courtesy of Carr et al. [57]**.**

Physically fibre surfaces can be considered as the bounding surface between two or more fields. The ability to visualise multiple scalars as fibre surfaces is seen as a tool to complement existing techniques, such as multivariate scatter plots [58], to allow quicker boundary extraction [57]. When compared to similar DVR techniques (Figure 4), such as multi-dimensional peak finding [59], it was found that fibre surfaces were quicker to render but had an addition pre-processing overhead. Fibre surfaces represent a simplified method of presenting the data captured using the multivariate topological techniques discussed in Section 5.

5. Multivariate topological visualisation

Multivariate topological visualisation is a more recent research interest in the visualisation community in comparison to topological visualisations of a single field. Due to the relative infancy of the field, Carr et al. [60] highlight the need for sufficiently complex data sets to extensively test the emerging algorithms in the area.

Many of the topological structures and algorithms applicable to a single variable can be generalised to more than one variable being defined at each sampling point. We begin this section by defining the concept of *Jacobi nodes*, we then continue to examine structures used to capture and the display the topology of multiple variables.

Jacobi node These are critical points in the scalar fields where the gradients of multiple inputs become parallel or equal to zero. A pre-requisite for the algorithms are that the fields are defined on a common sampling mesh meaning that the critical points are guaranteed to have the same geometric locations.

5.1 The Reeb space

The *Reeb space* is a generalisation of the Reeb graph to allow for multivariate or temporal data. The first discussion of using the Reeb space to compute topological structure of multiple functions is presented by Edelsbrunner et al. [61], where it is suggested that the Reeb space can be modelled mathematically in the form $f : \mathbb{M} \mapsto \mathbb{R}^k$, where $\mathbb{M}$ represents the domain and f the output of k scalar functions. For the simple case, where $k = 1$, this is directly comparable to the Reeb graph. The Reeb space extends this formulation to situations where $k \geq 2$.

5.2 The joint contour net

Carr et al. [62] presented the first discrete representation of the Reeb space using the *Joint Contour Net* (JCN). For functions of n variables defined in an $\mathbb{R}^m$ dimensional space the algorithm approximates the Reeb space as a number of multivariate contours named joint contour slabs. These represent connected regions of the domain with respect to the isovalue of multiple functions. In situations where $n \geq m$ the JCN can still be computed; however, the output is not an approximation of the Reeb space but instead a subdivision of the input geometry over n variables. The JCN captures the Reeb space as an undirected graph structure, where vertices represent slabs of n isovalue tuples, and edges are used to show adjacency between regions. An example JCN of two scalar functions is presented in Figure 5 and Figure 6.

In comparison to contour tree algorithms, the JCN is better suited to parallelisation as each joint contour slab is constructed from smaller independent regions called fragments [63]. Existing development of the algorithm has focused largely on implementation as a parallel algorithm. A distributed memory model is used by Duke and Hosseini [64] to construct multiple sub-JCNs in parallel, these are merged into a single output as a final post-processing step. Current results suggest that the merge step is the limiting factor to parallel algorithm speed up; therefore, other parallelisation strategies are under investigation.

In nuclear physics the JCN has previously been used to visualise and analyse scission datasets where it was used to identify the splitting of an atomic nucleus into multiple parts [65]. It was found that the JCN was well suited to capturing this divergent behaviour using proton and neutron density fields as inputs. This experiment was initially performed at a single temperature [66], but later repeated at multiple temperatures [67] due to its ability to capture the splitting of the compound nucleus as a forking in the multi-field topology. Whilst performing the analysis a number of other events were captured and linked to the scission theory.

More recently the JCN was used to visually analyse data from hurricane Isabel [68]. Vertices were used to represent the joint contour slabs by mapping to their barycentric spatial coordinates. An interactive environment was developed that allowed users to relate interactions in the temperature, pressure and precipitation fields to physical phenomena such as rain bands and the eye of the hurricane. The ability to relate properties of the JCN to known physical features helped to increase understanding of how the JCN is able to capture multi-field interactions.

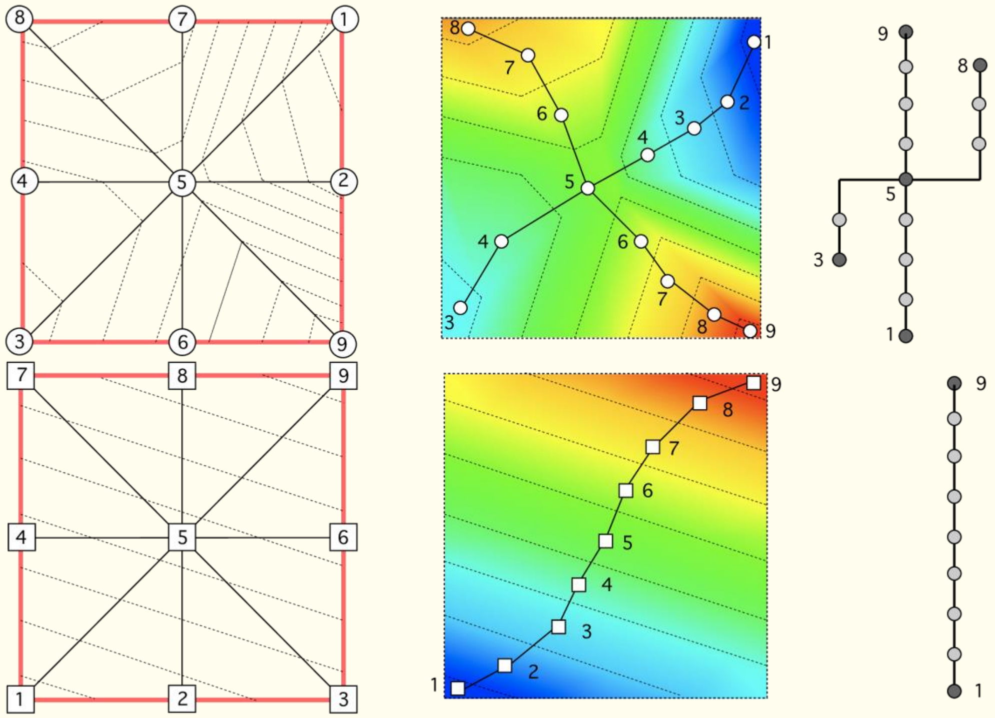

Figure 5 Two simple scalar functions are defined on a simplicial grid (left), where the dotted lines represent quantisation intervals. The quantised contour tree for each function (right) is shown mapped to the scalar field in the centre. Image courtesy of Duke et al. [65].

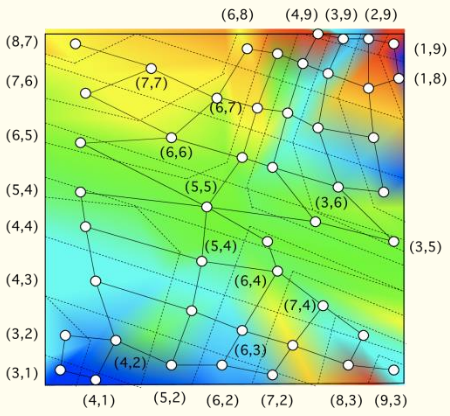

Figure 6 A JCN capturing the bivariate topology of the two simple functions shown in Figure 5. The bivariate field is constructed by overlaying the quantisation intervals of the two input fields (dotted lines). A vertex is placed at the barycentre of each region, or joint contour slab, and edges mark adjacency. Image courtesy of Duke et al. [65].

5.3 Related topological structures

In multivariate topology a *Jacobi set* represents the set of critical points generated when one Morse function is restricted to the level set of another. Alternatively, the Jacobi set can be considered as the set of points where there is an alignment of the gradient of two or more Morse functions or the gradient of one function vanishes. The similarity between two or more functions can be evaluated using the resulting Jacobi set using a method defined by Edelsbrunner et al. [69]. Applications of the Jacobi set include use as a feedback loop on simulation parameters or to perform comparison of algorithms. An algorithm was presented by Edelsbrunner and Harer [70] for computing Jacobi sets of multiple Morse functions defined on a triangular mesh of isovalue n-tuples in Euclidean space. This allows multiple independent functions such as pressure, temperature, or wind speed to be used as inputs. Alternatively for temporal data multiple time-steps can be used as input to allow the evaluation of a function with respect to time.

Carr et al. extend the JCN [62] to extract further topological structure from the Reeb space by first evaluating an intermediate structure, the *Multi-Dimensional Reeb graph* (MDRG) [71]. This is a hierarchical structure that recursively stores the joint contours of each function $(f_1, f_2, \ldots, f_n)$ restricted to the contours of those preceding it. At the top level the MDRG represents the contours of the function f_1 as a Reeb Graph; the second tier relates to the Reeb graph of function f_2 when restricted to the contours of f_1, continuing down to function f_n restricted to the contours of $f_1, \ldots, f_{n-1}$. Besides being used in the extraction of the Jacobi structure and the related *Reeb skeleton* [72], the MDRG represents a convenient structure for extracting the Reeb graphs of each individual function making up the Reeb space.

An additional abstraction, the *Jacobi structure*, related to the mathematical topology concept of *singular fibre-components* are introduced by Chattopadhyay et al. [71] as part of the MDRG extraction algorithm. This represents an improvement over the Jacobi set by providing a method of relating the Jacobi set directly to the Reeb space. The Jacobi structure is able to capture the exact location of topological change and is defined as the projection of the Jacobi set from the domain to the Reeb space. In practice this extends the Jacobi set to also include the "regular sheets" connecting one another in the Reeb space. This means the Jacobi structure is able to capture elements of the topological structure that the Jacobi set is unable to represent. The Jacobi structure is extracted as the set of critical nodes in the Multi-Dimensional Reeb Graph (MDRG), itself a structure for storing Reeb space criticalities. Forking in multi-field topology in nuclear scission data is an example behaviour that can be captured by the Jacobi structure [71].

The *Layered Reeb graph* is an alternative approach deployed by Strodthoff and Juttler [73] for presenting the Reeb space of multiple scalar functions. This approach to representing the Reeb space differs from the MDRG [71] by working directly with the Jacobi sets, rather than the more recently proposed Jacobi structure.

5.4 Topological persistence and simplification

Persistence in multivariate data sets is more complex to define in comparison to the univariate case. Simplification and persistence metrics can be defined on a number of secondary structures computable from the multi-field topology. The concept of isosurface statistics [48], [49] is extended to multivariate inputs through the use of *Continuous Scatterplots* [58]. These can be defined to show relations between m dimensional inputs with n scalar fields; in the case where $m = 3$ and $n = 1$ the output approximates to the output of Meyer et al. [49].

Multivariate data gives rise to *multi-filtrations* due to their parametrisation by more than one variable; this leads to no definable compact invariants, such as the Betti numbers, existing in multi-fields. Therefore, existing concepts, such as the persistence bar code [74], do not directly generalise to multi-variate domains. However, this does not mean that persistence and simplification cannot be applied, instead other approaches have been suggested. The "rank invariant" is a method for representing persistence in a multi-field by generalising upon the concept of Betti numbers present in univariate topology. For the univariate case, the rank invariant and persistence bar code are the same [75]. The original algorithm used to compute the rank invariant was exponential in time complexity, this was later improved to polynomial time by reformulating the problem as an algebraic geometry problem [76].

The Jacobi set, where the gradient of multiple functions align or have a gradient of zero, can assist in defining persistence measures [69]. When the multi-field is used to represent temporal data this can be used to augment the univariate notion of persistence with a lifetime parameter. This approach was used by Bremer et al. [77] to compute persistence in the context of the Morse-Smale complex. However, when generalised to non-temporal functions defining persistence as a feature of the Jacobi set becomes a non-trivial task [78].

5.5 The Reeb skeleton

The Reeb skeleton (Figure 7) is a simplified graph structure that takes into account the size of connected components, allowing measures of persistence to be assigned to its arcs. An extended Jacobi set, the *Jacobi Structure*, is used by the Reeb skeleton algorithm to aid multivariate simplification [72].

The Jacobi structure [71] is a promising starting point for simplification due to its ability to separate the Reeb space, as approximated by the JCN, into singular and *regular components*. Just as in the univariate equivalent, the Reeb graph, singular nodes in the Jacobi structure map to topological changes in the multi-field. To exploit this, the Reeb Skeleton extends the concept of the Jacobi structure further, primarily to aid multi-dimensional simplification [72].

The Reeb skeleton is generated as the dual graph of the singular and regular components captured in the Jacobi structure. Visually, this means the Reeb skeleton translates the sheet-like form of the Jacobi structure into a simplified skeletal form. The simplified graph data structure allows measures of persistence to be assigned to arcs of the Reeb skeleton in a similar manner to that of the Reeb graph. Lip pruning techniques, similar to the leaf pruning method of simplification found in univariate topological structures [18] can then be applied to progressively remove noisy features in the multi-field. Example persistence measures that can be applied to the JCN include the accumulated volume of joint contour slabs in a connected region.

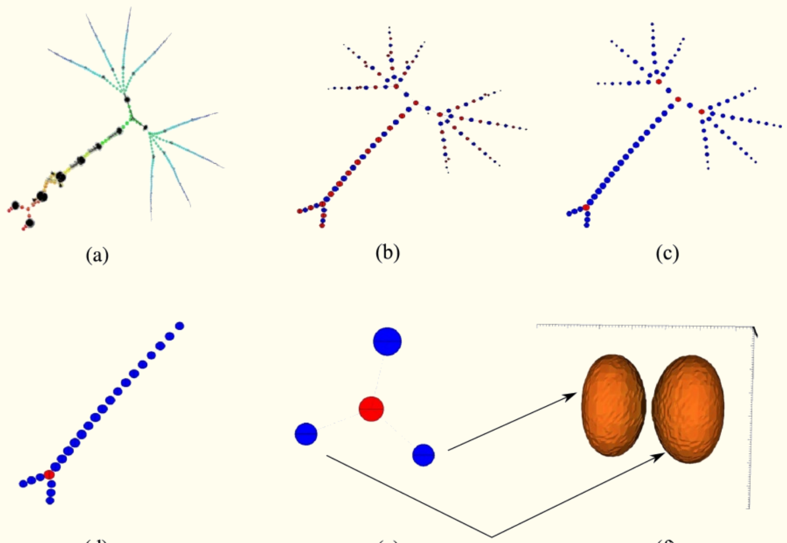

Figure 7 An example of a Reeb skeleton, showing how it relates to the JCN (a); (b) shows the full, non-simplified Reeb skeleton; (c) modifies the Reeb skeleton to highlight only critical changes in the multivariate topology (red vertices); (d) shows how performing simplification using the Reeb skeleton removes less significant regions of the topology; (e) after simplification, the Reeb skeleton can be reduced to only key vertices; (f) the arrowed regions relate to the two surfaces. Image courtesy of Chattopadhyay et al. [72].

6. Visualisation in higher dimensions

Name	Dimension n	Common name	Vertices	Edges	Faces	Cells
0-cube	0	Point	1	-	-	-
1-cube	1	Line-segment	2	1	-	-
2-cube	2	Square	4	4	1	-
3-cube	3	Cube	8	12	6	1
4-cube	4	Tesseract	16	32	24	8

Table 6 Properties of hypercubes in dimensions zero to four.

When moving to higher dimensional spaces it is beneficial to generalise the terminology used to describe the geometry. The n-dimensional analogue of the square is the *hypercube*, often shortened to n-cube (see Table 6). When working in higher dimensional spaces it is common to perform a simplicial sub-division of the n-cube into n-simplexes, this helps to avoid the ambiguities often associated with MC style algorithms [79].

6.1 Projection and perception

Creating easily perceivable and topologically correct three-dimensional models for projection on to flat surfaces, the computer monitor, is a difficult task. Volumes of data become increasingly hard to visualize as their dimensionality increase; for example, by introducing a temporal fourth-dimension. One of the major limiting factors is our in ability to perceive things four-dimensionally. A metaphor for trying to understand four-dimensions in a three-dimensional world is to consider the case of two-dimensional creatures trying to understand a three-dimensional world. This is a thought exercise discussed in [80], which also discusses how a four-dimensional Euclidean representation of space-time relates to real world physics.

Existing projection methods are available that take a four-dimensional objects and display them on a two-dimensional surface, usually in wire-frame form. Quite often the projections are animated to show the object as it rotates on one or more axis; however, this can also be adapted to allow the user to rotate the object through conventional approaches such as mouse interaction. The effect of perspective and isometric projection are explored by Hollasch [81] using tesseracts — the 4D hyper-cube. In addition, the use of ray-tracing in four-dimensions can produce understandable images, as depth cueing is handled automatically by the algorithm in the creation of shadows. However, the added complexity of a fourth dimension in a looping ray tracing algorithm makes it time consuming and potentially unworkable for real time display.

6.2 Computing surfaces in higher dimensions

Whilst presenting difficulties with regard to visualisation, it can be beneficial to compute surfaces in higher dimensions. This is especially true in the case of volumetric data with a temporal element; animation can often be used to reconstruct this form of data but that presents its own perceptual issues. Computation of isosurfaces on fields existing in $\mathbb{R}^4$ space can help to improve animations by providing interpolation between discrete time-steps, resulting in smoother and easier to perceive animations. Alternatively the high-dimensional topology can be sliced along arbitrary axes to provide a snap-shot with a reduction in dimension (e.g. $\mathbb{R}^4 \mapsto \mathbb{R}^3$).

An unavoidable consequence of upping the dimensionality of the input field is an increase in the complexity of its storage and computation. An early example of computing surfaces in > 3 dimensions is considered by Weigle and Banks [82] using a recursive technique to split n-dimensional cells into n-simplexes. The resulting surfaces exist in four dimensions and are able to be displayed using two techniques; stereographic projection (Section 6.1) or slicing to reduce dimensionality. It is noted that the centroid division technique used to break cells into simplexes used in this work is suboptimal (see Table 7), but can be improved using lookup tables similar to those used in marching cubes. The technique was used in [83] to compute the swept volumes of time-varying data generated from electromagnetic field simulations, allowing them to be displayed as animations.

n	Centroid division ($2^{n-1}n!$)	$n!$
2	4	2
3	24	6
4	192	24
5	1920	120
6	23040	720

Table 7 Number of simplexes generated using differing sub-division techniques in n dimensions.

Extending MC into the fourth dimension, and beyond, was explored by Bhaniramka et al. [84]. At the time of the report, a relatively small amount of work had been conducted in studying isosurfaces beyond the third dimension. It was found that by extending MC into 4 dimensions the look-up table, following removal of symmetrical configurations, required 222 separate configurations. As with the 3D variation of MC, one aspect to be taken into consideration is the dealing of ambiguous configurations; a mathematical proof of correctness is provided to verify that the topological structures generated are valid. An example use of this algorithm would be to present volume data with a time dimension by selecting three-dimensional slices of the hyper-volume.

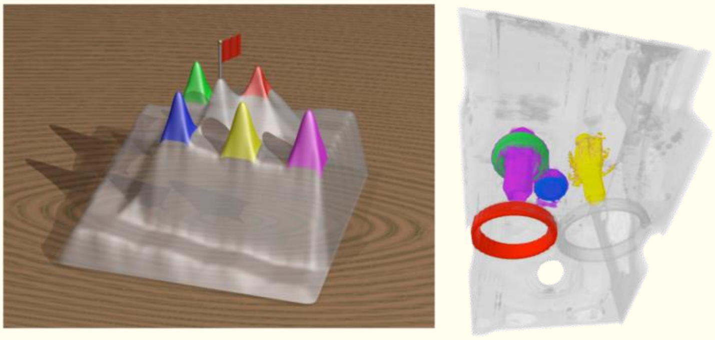

Figure 8 Peaks within a topological landscape (left) correspond to distinct topological features in the Direct Volume Render (right). Image courtesy of Weber et al. [51]**.**

6.3 Topological landscapes

An alternative approach of viewing contour information, the topological landscape, was proposed by Weber et al. [51], using the contour tree to build a 3D terrain model (Figure 8). The purpose of this is to harness the natural ability that humans have in understanding terrain structure and use it to provide an easier to understand model of the underlying data topology. Valleys in the terrain illustrate events where a contour splits into two or more parts and peaks represent where two or more contours merge. Topological landscapes can be applied to contour trees of any number of dimensions; hence, they can provide a useful method for exploring what is happening in high dimensional data sets. The topological landscape methodology was further expanded using a number of different methods for laying out the data, primarily from established 2D visualisation techniques [85].

7. Conclusion

In this chapter we discussed problems existing in marching cubes algorithms relating to topological correctness of the data. We demonstrated how, through the use of topology, a correct representation of a scalar field can be captured using graph structures. These could be used to seed topologically correct meshes for rendering, or to provide a means to analyse the data. After providing a description of algorithms for data sets consisting of a single variable, we showed how many of the techniques can be generalised to multivariate data. Finally, we considered the techniques for displaying data existing in higher dimensions.

Acknowledgments

This work used the resources of the DiRAC Facility jointly funded by STFC, the Large Facilities Capital Fund of BIS and Swansea University, and the DEISA Consortium (www.deisa.eu), funded through the EU FP7 project RI-222919, for support within the DEISA Extreme Computing Initiative. The work was also partly funded by EPSRC project: EP/M008959/1. The authors would also like to thank Dr. Hamish Carr for his advice and assistance throughout the duration of this work.

References

[1] W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," in *Computer Graphics (Proceedings of SIGGRAPH 87)*, 1987, vol. 21, no. 4, pp. 163–169.

[2] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel, "Feature sensitive surface extraction from volume data," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 57–66.

[3] G. M. Nielson, "Dual marching cubes," in *Proceedings of the conference on Visualization'04*, 2004, pp. 489–496.

[4] A. Bhattacharya and R. Wenger, "Constructing isosurfaces with sharp edges and corners using cube merging," in *Computer Graphics Forum*, 2013, vol. 32, no. 3pt1, pp. 11–20.

[5] T. S. Newman and H. Yi, "A survey of the marching cubes algorithm," *Comput. Graph.*, vol. 30, no. 5, pp. 854–879, 2006.

[6] Y. Zhou, W. Chen, and Z. Tang, "An elaborate ambiguity detection method for constructing isosurfaces within tetrahedral meshes," *Comput. Graph.*, vol. 19, no. 3, pp. 355–364, 1995.

[7] M. J. Dürst, "Re: additional reference to marching cubes," *ACM SIGGRAPH Comput. Graph.*, vol. 22, no. 5, p. 243, 1988.

[8] G. M. Nielson and B. Hamann, "The Asymptotic Decider: Removing the Ambiguity in Marching Cubes," in *Visualization '91*, 1991, pp. 83–91.

[9] E. V Chernyaev, "Marching cubes 33: Construction of topologically correct

isosurfaces," *Inst. High Energy Physics, Moscow, Russ. Rep. CN/95-17*, vol. 42, 1995.

[10] T. Lewiner, H. Lopes, A. W. Vieira, and G. Tavares, "Efficient implementation of marching cubes' cases with topological guarantees," *J. Graph. tools*, vol. 8, no. 2, pp. 1–15, 2003.

[11] T. Etiene *et al.*, "Topology verification for isosurface extraction," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 6, pp. 952–965, 2012.

[12] L. Custodio, T. Etiene, S. Pesco, and C. Silva, "Practical considerations on Marching Cubes 33 topological correctness," *Comput. Graph.*, vol. 37, no. 7, pp. 840–850, 2013.

[13] C. Montani, R. Scateni, and R. Scopigno, "A modified look-up table for implicit disambiguation of marching cubes," *Vis. Comput.*, vol. 10, no. 6, pp. 353–355, 1994.

[14] G. M. Nielson, "On marching cubes," *Vis. Comput. Graph. IEEE Trans.*, vol. 9, no. 3, pp. 283–297, 2003.

[15] H. Carr, "(No) more marching cubes," in *Proceedings of the Sixth Eurographics/Ieee VGTC conference on Volume Graphics*, 2007, pp. 81–90.

[16] H. Edelsbrunner, J. Harer, and A. Zomorodian, "Hierarchical Morse complexes for piecewise linear 2-manifolds," in *Proceedings of the seventeenth annual symposium on Computational geometry*, 2001, pp. 70–79.

[17] P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci, "A multi-resolution data structure for two-dimensional Morse-Smale functions," in *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, 2003, p. 19.

[18] H. Carr, J. Snoeyink, and M. van de Panne, "Simplifying flexible isosurfaces using local geometric measures," in *Proceedings of the conference on Visualization'04*, 2004, pp. 497–504.

[19] A. G. Landge *et al.*, "In-situ feature extraction of large scale combustion simulations using segmented merge trees," in *High Performance Computing, Networking, Storage and Analysis, SC14: International Conference for*, 2014, pp. 1020–1031.

[20] H. Edelsbrunner, D. Letscher, and A. Zomorodian, "Topological persistence and simplification," *Discret. Comput. Geom.*, vol. 28, no. 4, pp. 511–533, 2002.

[21] H. Carr, J. Snoeyink, and U. Axen, "Computing contour trees in all dimensions," *Comput. Geom.*, vol. 24, pp. 75–94, 2003.

[22] R. L. Boyell and H. Ruston, "Hybrid techniques for real-time radar simulation," in *Proceedings of the November 12-14, 1963, fall joint computer conference*, 1963, pp. 445–458.

[23] T. Banchoff and others, "Critical points and curvature for embedded polyhedra," *J. Differ. Geom.*, vol. 1, no. 3–4, pp. 245–256, 1967.

[24] H. Freeman and S. P. Morse, "On searching a contour map for a given terrain elevation profile," *J. Franklin Inst.*, vol. 284, no. 1, pp. 1–25, 1967.

[25] V. Pascucci and K. Cole-McLaughlin, "Efficient computation of the topology of level

sets," in *Proceedings of the conference on Visualization'02*, 2002, pp. 187–194.

[26] M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore, "Contour trees and small seed sets for isosurface traversal," *Proc. Thirteen. Annu. Symp. Comput. Geom. - SCG '97*, pp. 212–220, 1997.

[27] S. P. Tarasov and M. N. Vyalyi, "Construction of contour trees in 3D in O (n log n) steps," in *Proceedings of the fourteenth annual symposium on Computational geometry*, 1998, pp. 68–75.

[28] H. Carr and M. Van Panne, "Topological manipulation of isosurfaces," The University of British Columbia (Canada), 2004.

[29] H. Carr and J. Snoeyink, "Path seeds and flexible isosurfaces using topology for exploratory visualization," in *Proceedings of the symposium on Data visualisation 2003*, 2003, pp. 49–58.

[30] M. De Berg and M. van Kreveld, "Trekking in the Alps Without Freezing or Getting Tired," *Algorithmica*, vol. 18. pp. 306–323, 1997.

[31] Y.-J. Chiang, T. Lenz, X. Lu, and G. Rote, "Simple and optimal output-sensitive construction of contour trees using monotone paths," *Comput. Geom.*, vol. 30, no. 2, pp. 165–195, 2005.

[32] S. Takahashi, T. Ikeda, Y. Shinagawa, T. L. Kunii, and M. Ueda, "Algorithms for extracting correct critical points and constructing topological graphs from discrete geographical elevation data," in *Computer Graphics Forum*, 1995, vol. 14, no. 3, pp. 181–192.

[33] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny, "Contour Forests: Fast Multi-threaded Augmented Contour Trees," 2016.

[34] G. Reeb, "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. Sci. Paris*, vol. 222, pp. 847–849, 1946.

[35] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci, "Loops in Reeb graphs of 2-manifolds," in *Proceedings of the nineteenth annual symposium on Computational geometry*, 2003, pp. 344–350.

[36] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas, "Robust on-line computation of Reeb graphs: simplicity and speed," in *ACM Transactions on Graphics (TOG)*, 2007, vol. 26, no. 3, p. 58.

[37] I. Fujishiro, Y. Takeshima, T. Azuma, and S. Takahashi, "Volume data mining using 3D field topology analysis," *IEEE Comput. Graph. Appl.*, no. 5, pp. 46–51, 2000.

[38] Y. Shinagawa and T. L. Kunii, "Constructing a Reeb graph automatically from cross sections," *IEEE Comput. Graph. Appl.*, no. 6, pp. 44–51, 1991.

[39] Y. Shinagawa, T. L. Kunii, and Y. L. Kergosien, "Surface coding based on Morse theory," *IEEE Comput. Graph. Appl.*, no. 5, pp. 66–78, 1991.

[40] W. Harvey, Y. Wang, and R. Wenger, "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*, 2010, pp. 267–276.

[41] J. Tierny, A. Gyulassy, E. Simon, and V. Pascucci, "Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees," *Vis. Comput. Graph. IEEE Trans.*, vol. 15, no. 6, pp. 1177–1184, 2009.

[42] H. Doraiswamy and V. Natarajan, "Efficient algorithms for computing Reeb graphs," *Comput. Geom.*, vol. 42, no. 6, pp. 606–616, 2009.

[43] Doraiswamy, Harish and Natarajan, Vijay, H. Doraiswamy, and V. Natarajan, "Output-sensitive construction of Reeb graphs," *Vis. Comput. Graph. IEEE Trans.*, vol. 18, no. 1, pp. 146–159, 2012.

[44] H. Doraiswamy and V. Natarajan, "Computing Reeb graphs as a union of contour trees," *Vis. Comput. Graph. IEEE Trans.*, vol. 19, no. 2, pp. 249–262, 2013.

[45] G. Wyvill, C. McPheeters, and B. Wyvill, "Data structure for soft objects," *Vis. Comput.*, vol. 2, no. 4, pp. 227–234, 1986.

[46] H. Carr, J. Snoeyink, and M. van de Panne, "Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree," *Comput. Geom.*, vol. 43, no. 1, pp. 42–58, 2010.

[47] C. L. Bajaj, V. Pascucci, and D. R. Schikore, "The contour spectrum," in *Proceedings of the 8th conference on Visualization'97*, 1997, pp. 167--173.

[48] H. Carr, D. Brian, and D. Brian, "On histograms and isosurface statistics," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 5, pp. 1259–1266, 2006.

[49] M. Meyer, C. E. Scheidegger, J. M. Schreiner, B. Duffy, H. Carr, and C. T. Silva, "Revisiting histograms and isosurface statistics," *IEEE Trans. Vis. Comput. Graph.*, vol. 14, no. 6, pp. 1659–1666, 2008.

[50] Y. Takeshima, S. Takahashi, I. Fujishiro, and G. M. Nielson, "Introducing topological attributes for objective-based visualization of simulated datasets," in *Volume Graphics, 2005. Fourth International Workshop on*, 2005, pp. 137–236.

[51] G. H. Weber, P.-T. Bremer, and V. Pascucci, "Topological landscapes: A terrain metaphor for scientific data," *Vis. Comput. Graph. IEEE Trans.*, vol. 13, no. 6, pp. 1416–1423, 2007.

[52] K. Beketayev, D. Yeliussizov, D. Morozov, G. H. Weber, and B. Hamann, "Measuring the distance between merge trees," in *Topological Methods in Data Analysis and Visualization III*, Springer, 2014, pp. 151–165.

[53] H. Bunke and K. Riesen, "Graph edit distance: optimal and suboptimal algorithms with applications," *Anal. complex networks, from Biol. to Linguist.*, pp. 113–143, 2009.

[54] H. Saikia, H.-P. Seidel, and T. Weinkauf, "Extended Branch Decomposition Graphs: Structural Comparison of Scalar Data," *Comput. Graph. Forum*, vol. 33, no. 3, pp. 41–50, 2014.

[55] D. M. Thomas and V. Natarajan, "Symmetry in scalar field topology," *Vis. Comput. Graph. IEEE Trans.*, vol. 17, no. 12, pp. 2035–2044, 2011.

[56] D. Laney, P.-T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci, "Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities," *Vis. Comput. Graph. IEEE Trans.*, vol. 12, no. 5, pp. 1053–1060, 2006.

[57] H. Carr, G. Zhao, J. Tierny, A. Chattopadhyay, and A. Knoll, "Fiber Surfaces: Generalizing Isosurfaces to Bivariate Data," in *Computer Graphics and Visual Computing (CGVC) 2015*, 2015, pp. 63–64.

[58] S. Bachthaler and D. Weiskopf, "Continuous scatterplots," *IEEE Trans. Vis. Comput. Graph.*, vol. 14, no. 6, pp. 1428–1435, 2008.

[59] N. Kotava *et al.*, "Volume rendering with multidimensional peak finding," in *Visualization Symposium (PacificVis), 2012 IEEE Pacific*, 2012, pp. 161–168.

[60] H. A. Carr, J. Tierny, and G. H. Weber, "Pathological and Test Cases For Reeb Analysis," in *Topology-Based Methods in Visualization 2017 (TopoInVis 2017)*, 2017.

[61] H. Edelsbrunner, J. Harer, and A. K. Patel, "Reeb spaces of piecewise linear mappings," in *Proceedings of the twenty-fourth annual symposium on Computational geometry*, 2008, pp. 242–250.

[62] H. Carr and D. Duke, "Joint Contour Nets," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 8, pp. 1100–1113, 2013.

[63] D. J. Duke, F. Hosseini, and H. Carr, "Parallel computation of multifield topology: experience of Haskell in a computational science application," in *Proceedings of the 3rd ACM SIGPLAN workshop on Functional high-performance computing*, 2014, pp. 11–21.

[64] D. J. Duke and F. Hosseini, "Skeletons for distributed topological computation," in *Proceedings of the 4th ACM SIGPLAN Workshop on Functional High-Performance Computing*, 2015, pp. 35–44.

[65] D. Duke, H. Carr, A. Knoll, N. Schunck, H. A. Nam, and A. Staszczak, "Visualizing nuclear scission through a multifield extension of topological analysis," *Vis. Comput. Graph. IEEE Trans.*, vol. 18, no. 12, pp. 2033–2040, 2012.

[66] N. Schunck, D. Duke, H. Carr, and A. Knoll, "Description of induced nuclear fission with Skyrme energy functionals: Static potential energy surfaces and fission fragment properties," *Phys. Rev. C*, vol. 90, no. 5, p. 54305, 2014.

[67] N. Schunck, D. Duke, and H. Carr, "Description of induced nuclear fission with Skyrme energy functionals. II. Finite temperature effects," *Phys. Rev. C*, vol. 91, no. 3, p. 34327, 2015.

[68] Z. Geng, D. Duke, H. Carr, and A. Chattopadhyay, "Visual analysis of hurricane data using joint contour net," in *Computer Graphics and Visual Computing (CGVC)*, 2014.

[69] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci, "Local and global

comparison of continuous functions," in *Visualization, 2004. IEEE*, 2004, pp. 275–280.

[70] H. Edelsbrunner and J. Harer, "Jacobi sets of multiple Morse functions," *Found. Comput. Math. Minneap.*, vol. 8, pp. 37–57, 2002.

[71] A. Chattopadhyay, H. Carr, D. Duke, and Z. Geng, "Extracting Jacobi Structures in Reeb Spaces," in *EuroVis-Short Papers*, 2014.

[72] A. Chattopadhyay, H. Carr, D. Duke, Z. Geng, and O. Saeki, "Multivariate topology simplification," *arXiv Prepr. arXiv1509.04465*, 2015.

[73] B. Strodthoff and B. Jüttler, "Layered Reeb graphs of a spatial domain," *Bookl. Abstr. EuroCG*, pp. 21–24, 2013.

[74] A. Zomorodian and G. Carlsson, "Computing persistent homology," *Discrete Comput. Geom.*, vol. 33, no. 2, pp. 249–274, 2005.

[75] G. Carlsson and A. Zomorodian, "The theory of multidimensional persistence," *Discrete Comput. Geom.*, vol. 42, no. 1, pp. 71–93, 2009.

[76] G. Carlsson, G. Singh, and A. Zomorodian, "Computing multidimensional persistence," in *Algorithms and computation*, Springer, 2009, pp. 730–739.

[77] P. T. Bremer *et al.*, "Topological feature extraction and tracking," in *Journal of Physics: Conference Series*, 2007, vol. 78, no. 1, p. 12007.

[78] N. Suthambhara and V. Natarajan, "Simplification of jacobi sets," in *Topological Methods in Data Analysis and Visualization*, Springer, 2011, pp. 91–102.

[79] G. M. Nielson and B. Hamann, "The asymptotic decider: resolving the ambiguity in marching cubes," in *Proceedings of the 2nd conference on Visualization'91*, 1991, pp. 83–91.

[80] R. von Bitter Rucker, *Geometry, relativity and the fourth dimension*. Dover Publications, 1977.

[81] S. R. Hollasch, "Four-Space Visualization of 4D Objects," Arizona State University, 1991.

[82] C. Weigle and D. C. Banks, "Complex-valued contour meshing," in *Proceedings of the 7th conference on Visualization'96*, 1996, p. 173--ff.

[83] C. Weigle and D. C. Banks, "Extracting iso-valued features in 4-dimensional scalar fields," in *Proceedings of the 1998 IEEE symposium on Volume visualization*, 1998, pp. 103–110.

[84] P. Bhaniramka, R. Wenger, and R. Crawfis, "Isosurfacing in higher dimensions," *Proc. Vis. 2000. VIS 2000 (Cat. No.00CH37145)*, 2000.

[85] W. Harvey and Y. Wang, "Topological landscape ensembles for visualization of scalar-valued functions," *Comput. Graph. Forum*, vol. 29, pp. 993–1002, 2010.