Tracking change in Topology during Downsampling

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFIMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

Master of Engineering

IN

Computer Science And Engineering

BY

Raman Preet Kaur

Computer Science and Automation

Indian Institute of Science

Bangalore – 560 012 (INDIA)

June, 2016

Declaration of Originality

I, **Raman Preet Kaur**, with SR No. **04-04-00-10-41-14-1-11145** hereby declare that the material presented in the thesis titled

Tracking change in Topology during Downsampling

represents original work carried out by me in the **Department of Computer Science and Automation** at **Indian Institute of Science** during the years **2014-16**.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: Prof. Vijay Natarajan Advisor Signature

DEDICATED TO

The Student Community

Acknowledgements

I express my profound gratitude and sincere thanks to Prof. Vijay Natarajan for his valuable guidance, caring supervision and priceless feedback for the progress of this project. I am grateful to everyone who stood beside me during the course of my project. I would like to express my sincere gratitude to all my labmates especially Vidya and Talha for their help and support, throughout the duration of my project. I would like to thank all my classmates and colleagues for their unconditional support throughout my course.

Abstract

We study the topological effects of downsampling in 3-D scalar fields represented over structured grids. To capture the extent of topological changes during downsampling, we consider Extended Branch Decomposition Graph (eBDG) based measure. While computing eBDG score of comparison between low resolution scalar fields and original scalar field, an additional parameter of position along with persistence is introduced. We compare and analyze the eBDG cost with Root Mean Sqaure Distance (RMSD) and Structural SIMilarity (SSIM) index.

Contents

List of Figures

Chapter 1

Introduction

Scientific simulations and experiments such as study of objects or spaces which use scanning techniques like satellite imaging, medical imaging, 3D radar imaging, and various fields in physics where we need temperature distribution throughout space, pressure distribution in a fluid often generate scalar fields. These scalar fields are available as an array of values over the polygonal mesh (structured or unstructured). For space or low resolution requirements various applications require to downsample the structured or unstructured grids. Downsampling is common in image processing, scientific visualization of data in fields ranging from material science to medicine. But downsampling can introduce geometric or topological errors. Topological properties in scalar fields carry important information. These help to navigate in data, identify noise, used to compare, analyze and simplify volumetric data, check the quality and plausibility of extracted shapes and structures [3,6,13,18,19,24], so analyzing the change in topology while downsampling becomes critical.

The study aims to track and analyze the change in topology in 3D structured grids of scalar values as we move from high resolution data to low resolution data during downsampling. It helps to analyze various downsampling techniques by relatively comparing the topological changes introduced by each approach. This study also helps to visualize which topological properties/features are retained or lost during downsampling. Our work helped to quantify the extent of topological change introduced during downsampling.

Chapter 2

Motivation

Some work has been done in the field of topology aware or topology preserving dowsampling [4,10]. But none of them gives a guarantee to preserve topology. So we first study the standard sampling techniques and aim to capture the topology change during iterative downsampling. We analyze and quantify the change in topology at each iteration, which can help the user to stop at certain iteration where the topology change exceeds the permissible limit.

Also most of the applications require to retain the type of mesh during downsampling. The topology aware or topology preserving downsampling techniques apply mesh modifications. In 3D these techniques decompose the original mesh into tetrahedral mesh. Bong-Soo and Sohn [17] have shown topology preserving tetrahedral decomposition of trilinear cell where they decompose a cube with trilinear interpolation into a set of tetrahedra with linear interpolation, where they are trying to preserve isosurface topology. So modifying the original mesh type is an additional constraint imposed by topology aware downsamping techniques. But in our study we are using standard sampling techniques without modifying the mesh.

Further there is work done to capture the topology changes in unstructured grids but structured grids which are more ubiquitous need to be studied for the topological effects of downsampling.

Chapter 3

Background

The work builds upon methods developed that apply Morse theory to computer graphics and visualization. Morse theory studies the topology of manifolds by way of level sets of scalar fields defined over the manifolds. The topological properties of a scalar field refer to the topology of the level sets.
Let $f : \mathbb{M} \longrightarrow \mathbb{R}$ be a piecewise linear function defined over a mesh $\mathbb{M}$. Let $\mathbb{M}$ be a triangulated mesh. For a vertex v of $\mathbb{M}$, the *star* of v, denoted by $\mathrm{St}(v)$, consists of all triangles incident on v. The **link** of v, denoted by $\mathrm{Lk}(v)$, is the boundary of $\mathrm{St}(v)$ i.e., the cycle formed by edges of $\mathbb{M}$ that are not incident on v but belong to the triangles of $\mathrm{St}(v)$. The lower(resp. upper) link of v, Lk^{-}(v)(resp. $\mathrm{Lk}^{+}(v)$), is the subgraph of $\mathrm{Lk}(v)$ induced by vertices u with $h(u) < h(v)$(resp. $f(u) < f(v)$) [2].
A **minimum**(resp. **maximum**) of $\mathbb{M}$ is a vertex v for which $\mathrm{Lk}^{+}(v)(resp. Lk^{-}(v))$ is empty. A maximum or a minimum vertex is called a *extremal* vertex. A non-extremal vertex v is regular if $Lk^{-}(v)$ (and also $Lk^{+}(v)$) is connected, and saddle otherwise.
The **level set** for an isovalue $v \in \mathbb{R}$ is defined as a set of points $p \in \mathbb{M}$ where $f(p) = v$. The v-sublevel set and v-superlevel set of $\mathbb{M}$ denoted by $\mathbb{M}_{<v}$ and $\mathbb{M}_{>v}$ respectively, consist of points $p \in \mathbb{M}$ with $f(p) < v$ and $f(p) > v$ respectively. We refer to level set $\mathbb{M}_v$ where $v = f(p)$ for some critical vertex p as a critical level. A *contour* of $\mathbb{M}$ is a connected component of a level set of $\mathbb{M}$.
Contour tree is an abstract way of representing a scalar field. It was introduced by Boyell and Ruston [4] as a summary of the evolution of contours on a map(i.e. in 2-D). Contour tree was used by Freeman and Morse to find terrain profiles in a contour map [7]. This was then introduced by Van Kreveld et al. [22] to compute isolines on terrain maps in geographic information systems. It captures the topological properties of a scalar field. The contour tree is a graph that tracks contours of the level set as they split, join, appear and disappear.

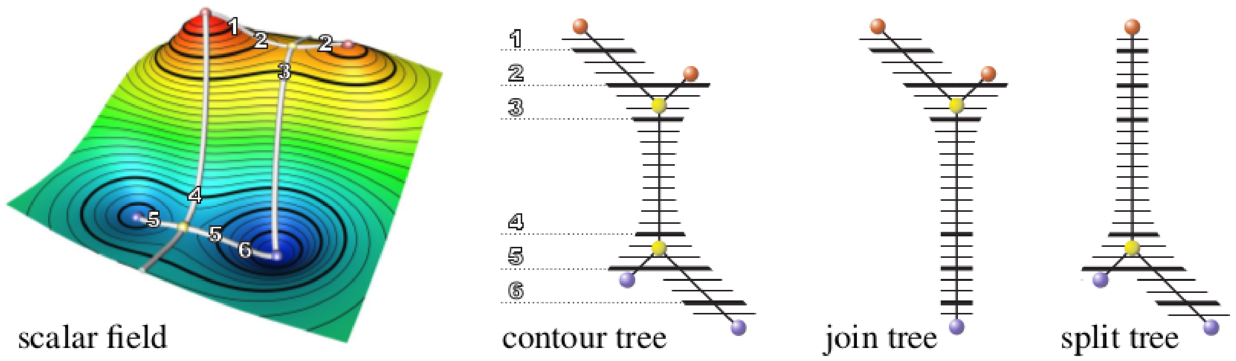


Figure 3.1: Contour tree tracks the contours of the level set as they split, join, appear and disappear. It can be created by merging the join tree and split tree. Adapted from [15].

It is obtained by contracting level set to a point. Formally, it is the quotient space under an equivalence relation that identifies all points within a connected component of a level set. Intuitively it is obtained by continuously collapsing each contour in the level set into a single point. One can imagine that for continuous function there is surjection map $\phi : \mathbb{M} \longrightarrow \mathbb{T}_f$ such that $\phi(x) = \phi(y)$ if and only if x and y are from the same contour. To obtain the contour tree, sweep through the input in decreasing order of function values. A vertex is a *join saddle* if two level set components merge at that vertex during the sweep. It is *split saddle* if a single component splits into two components at that vertex. The method is explained in Figure 3.1 [15]. The contour tree has been extensively used in image processing and geographic information systems. [9,11,16,18]. Contour trees are unrooted and can be computed by merging two rooted trees: the *join* tree and *split* tree, together referred as *merge* trees.
A join tree can be subdivided hierarchically into branches known as **branch decomposition tree (BDT)** according to the weight of the edges. Each node in BDT corresponds to branch of the join tree commenced by Pascucci et al. [14] shown in Figure 3.2 [15].

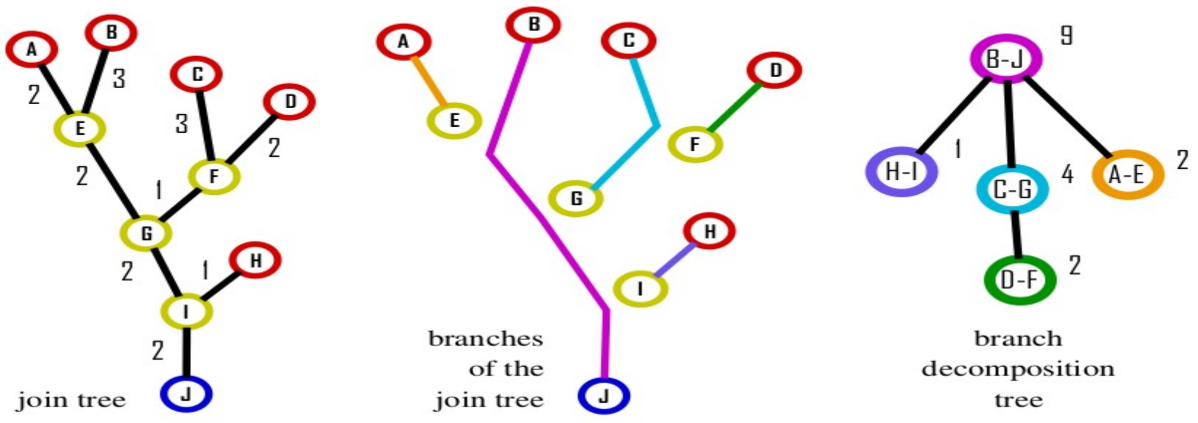


Figure 3.2: Branch Decomposition Tree. Adapted from [15].

Chapter 4

Related Work

Martin Kraus et al. [10] studied topology guided downsampling for structured volume grids. They have tried to preserve the scalar values of the critical points and not the critical points themselves. It is quite possible that the critical point might shift to the point in the neighborhood. And they have also assumed the relative importance of critical points to preserve the scalar values i.e. extremal points in their study have been given more importance than saddles and many more.
Renato Pajarola et al. [8] studied topology preserving and controlled topology simplifying multiresolution isosurface extraction based on recursive bisection of tetrahedra. There they modify the mesh and also they give no guarantee of topology being preserved.
Bong-Soo et al. [17] studied topology preserving tetrahedral decomposition of trilinear cell. Again there they give no guarantee for topology preservation.
There is work done on volume rendering of large scalar fields on low resolution devices (mobile devices) using subsampling techniques [20] where they propose sampling techniques that help to preserve topology during subsampling. But there too it is not guaranteed that the topology is preserved.

There exist approaches that deal with the comparison of topological structures. Beketayev et al. [12] compare two merge trees by means of branch decompositions. Large number of branch decompositions are considered to avoid instabilities resulting in long computation times. Cohen-Steiner at al. [5] introduced bottleneck distance between persistence diagrams. The measure is robust to noise but does not incorporate sub-level set nesting information. The sub-level set information is better captured by merge trees. The closest work to our study is due to Saikia et al. [15]. They compare all subtrees of two merge trees in an efficient manner exploiting redundancy. They introduce a novel data structure called the extended branch decomposition graph (eBDG) which is composed of the branch decompositions of all subtrees of the merge tree. An efficient algorithm based on dynamic programming has been provided to compare two eBDGs. Our study extends their work to include position of critical points along with the persistence to compute the comparison score.

Chapter 5

Methods

For tracking and quantifying the change in topology during downsampling, the idea is to use eBDG based measure.

eBDG Computation and Comparison

1. Computation:

The first step is to compute the join trees for original scalar field and downsampled scalar fields. The join trees are computed using a fast and memory efficient parallel algorithm proposed in [1] with time complexity of $O(V_{grid} + T \log T)$ where V_{grid} are the vertices in the structured grid and T are the critical points in the complete domain. The join tree is represented by (V, E) where V represents the critical points and $V \subseteq T$ as it includes maxima, saddles and one global minimum and each edge is given by (v_i, v_j) such that $v_i, v_j \in V$.

The join trees obtained are used to compute the eBDGs. eBDG avoids redundancy by combining the branch decomposition trees (BDT) of all the subtrees of a join tree into a single graph structure. The **branch decomposition tree** B_{bdt_i} is given by:

$$B_{bdt\_i} = (B_i, S_i)$$

with the nodes $B_i = \{b : b = \{e_x, ..., e_y\} \text{ with } e_x, ..., e_y \in E\}$ representing the branches and the edges S_i representing their hierarchy.

For **eBDG computation** we start with edges of the join tree that contain a maximum. The branch decomposition tree for these edges will be a single node representing the edge. The position P_i corresponding to these eBDG nodes will be the position of associated maximum. We initialize the eBDG with these trivial branch decompositions B_{bdt_i} for all maxima $v_i \in V$:

$$B_{bdt\_i} = (e_i, \{\})$$

where $e_i = (v_i \to v_j) = (v_i, v_j)$ is the unique edge in the join tree starting from v_i. The weights

W_i of these trivially created nodes B_{bdt_i} and their levels L_i are given by:

$$W_i = w_i \qquad L_i = 0$$

where w_i is the weight of the edge e_i. This initialization creates the level 0 nodes of the eBDG. Then the algorithm visits every saddle of the join tree and updates the eBDG. The update of each saddle consists of adding a new node and number of edges to the eBDG.

The updation is done as follows:

Let v_i denotes a saddle with $(v_k \rightarrow v_i) \in E$ and all B_{bdt_k} have already been computed. Then we can compute $B_{bdt\_i} = (B_i, S_i)$ as follows:

$$l = argmax\{W_x : (v_x \rightarrow v_i) \in E\}$$
$$B_i = e_i \cup B_l$$
$$S_i = S_l \cup \{(B_i \rightarrow B_k) : (v_k \rightarrow v_i) \in E \wedge k \neq l\}$$

with the weight of the root node of B_{bdt_i}, level and position given by

$$W_i = W_l + w_i$$
$$L_i = max(\{1 + L_k : (v_k \rightarrow v_i) \in E \wedge k \neq l\} \cup L_l)$$

P_i is the position of the maximum associated with the root node of B_{bdt_l}.

This computation at every saddle v_i can be interpreted as: the heaviest branch B_l is identified and combined with the edge e_i to form root node of B_{bdt_i}. All other branches are assigned as children. Also all children of heaviest branch are assigned as children of B_{bdt_i}.

2. Comparison:

After computing the eBDGs for original and downsampled versions, the next step would be the comparison of the eBDGs. Bottom-up approach and memoization are used to compare an eBDG node to another - including their children.

Consider two eBDGs B^1 and B^2 where B^1 represents the eBDG computed for original dataset and B^2 represents eBDG computed for the downsampled version.

The cost between the **level-**0 nodes B^1_i and B^2_j is given by the cost function:

$$c(B^1_i, B^2_j) = \frac{dist(P^1_i, P^2_j) + \mid W^1_i - W^2_j \mid}{2}$$

where $dist(P^1_i, P^2_j)$ is the L^2 norm or L^2 distance between P^1_i and P^2_j.

Higher level nodes have children which also need to be matched up such that it leads to the minimum cost. The key of the algorithm is memoization: if the minimum cost of matching a lower-level node is already computed, then it requires to match the immediate children.

The eBDG nodes are visited in ascending order of the levels and compute the cost as:

$$c(B_i^1, B_j^2) = \frac{dist(P_i^1, P_j^2) + \mid W_i^1 - W_j^2 \mid}{2} + c(F_i^1, F_j^2)$$

where $c(F_i^1, F_j^2)$ is the cost of matching the two sequences of trees below B_i^1 and B_j^2. The lower level are processed first, then this only requires to match up the sequences of values. The matching obtained must not have any crossings because if present, that implies higher valued saddles are getting matched to lower-valued saddles.
Matching two sequences without edge crossings can be solved by divide-and-conquer as proposed in [8]. There are three choices at any moment:

1. leave out the first node in the first sequence.
2. leave out the first node in the second sequence.
3. match the first two nodes.

Then this leads to following formulation,

$$c(F_i^1, F_j^2) = d_{0,0}(F_i^1, F_j^2)$$

where $d_{p,q}(F_i^1, F_j^2)$ is the minimal cost of matching the two sequences of trees F_i^1 and F_j^2 at indices p and q respectively.
F_i^1 and F_j^2 are given by:

$$F_i^1 = \{B_{i,0}^1, B_{i,1}^1, B_{i,2}^1, ...B_{i,m}^1\}$$

$$F_j^2 = \{B_{j,0}^2, B_{j,1}^2, B_{j,2}^2, ...B_{j,n}^2\}$$

Here B_i^1 has m children and B_j^2 has n children. The $d_{p,q}(\{B_{i,0}^1, B_{i,1}^1, B_{i,2}^1, ...B_{i,m}^1\}, \{B_{j,0}^1, B_{j,1}^1, B_{j,2}^1, ...B_{j,n}^1\})$ can be written as $d_{p,q}$. The **dynamic programming expression** can be written as:

$$d_{p,q} = min \begin{cases} d_{p+1,q} + c(B_{i,p}^1) \\ d_{p,q+1} + c(B_{j,q}^2) \\ d_{p+1,q+1} + c(B_{i,p}^1, B_{j,q}^2) \end{cases}$$

where $p \in \{0, ..., m\}$ and $q \in \{0, ..., n\}$. The $d_{p,n+1} = d_{m+1,q} = 0$. Whenever a node is left, the cost of leaving that node also includes the cost of leaving all its children.
That can be computed as:

$$c(B_i) = W_i + c(F_i)$$
$$c(F_i) = \sum c(B_k)\ \forall k \in children(i)$$

Thus an all to all score matrix is obtained. The rows and columns of the matrix correspond to the nodes of B^1 and B^2 respectively. Now from this matrix we find the minimum score column for the last row. The last row corresponds to the node of B^1 which maps to complete join tree of the original scalar field. We backtrack and find the matchings for this node and its children. The cost of comparison (score) and the matched and unmatched nodes are analysed and plotted as graphs and visualized in **paraview** (an open source multiple-platform application for interactive scientific visualization).
The space complexity of the comparison algorithm is $O(N_1N_2 + b_1b_2)$ where N_1 and N_2 denote the sizes of the two eBDGs and b_1 and b_2 denote the average branching factors. The time complexity is $O(N_1N_2b_1b_2)$.

<u>RMSD</u>

We compute the RMSD values for the downsampled versions on comparing with the original scalar field. The downsampled versions are tri-linearly interpolated to obtain the same number of points in the original scalar field.
RMSD is given by:

$$\text{RMSD(X,Y)} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}|x_i - y_i|^2}$$

where X,Y are the original and interpolated downsampled scalar fields respectively.

<u>SSIM</u>

The third measure computed is SSIM, proposed by Wang et al. [17]. SSIM is a method to quantify the visibility of errors (differences) between a distorted image and a reference image. In our study we have extended it to 3D so as to use the same for the scalar fields defined on 3D structured grids. To compute SSIM, similar to RMSD, the downsampled scalar fields are tri-linarly interpolated. Suppose x and y are two set of scalar values on a subset of points of 3D structured grid (e.g., spatial subcubes extracted from original and downsampled scalar field). Here at each point, local statistics μ_x, σ_x and σ_{xy} are computed. Here μ_x is mean intensity, σ_x

is standard deviation and σ_{xy} is the correlation (inner product) cofficient between x and y, all are considered in a simple and effective way to quantify structural similarity.

SSIM is given by:

$$\text{SSIM}(x,y)=\frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)}$$

The constants C_1 and C_2 are introduced to avoid instability when denominator is close to zero.

If X,Y are the original and interpolated downsampled scalar fields respectively; x_j and y_j are the scalar field values at the jth local window ($11\times11\times11$ cube centered at jth grid point); M is the number of local windows of the scalar fields, then the Mean SSIM (MSSIM) is computed as:

$$\text{MSSIM(X,Y)} = \frac{1}{M}\sum_{j=1}^{M}\text{SSIM}(x_j, y_j)$$

The implementation and results are performed on Intel Core i5-3337U CPU 1.80GHz$\times$4 processor with 4GB RAM and 100GB hard disk.

Chapter 6

Results

To compare and analyze the change in topology during downsampling, the experiment is performed on **Fuel dataset** (obtained from volvis.org).
The original fuel dataset shown in **Figure 6.1** has the resolution $64 \times 64 \times 64$, storing 8 bits per voxel. The dataset is simulation of fuel injection into a combustion chamber. The higher the density value, the less presence of air.

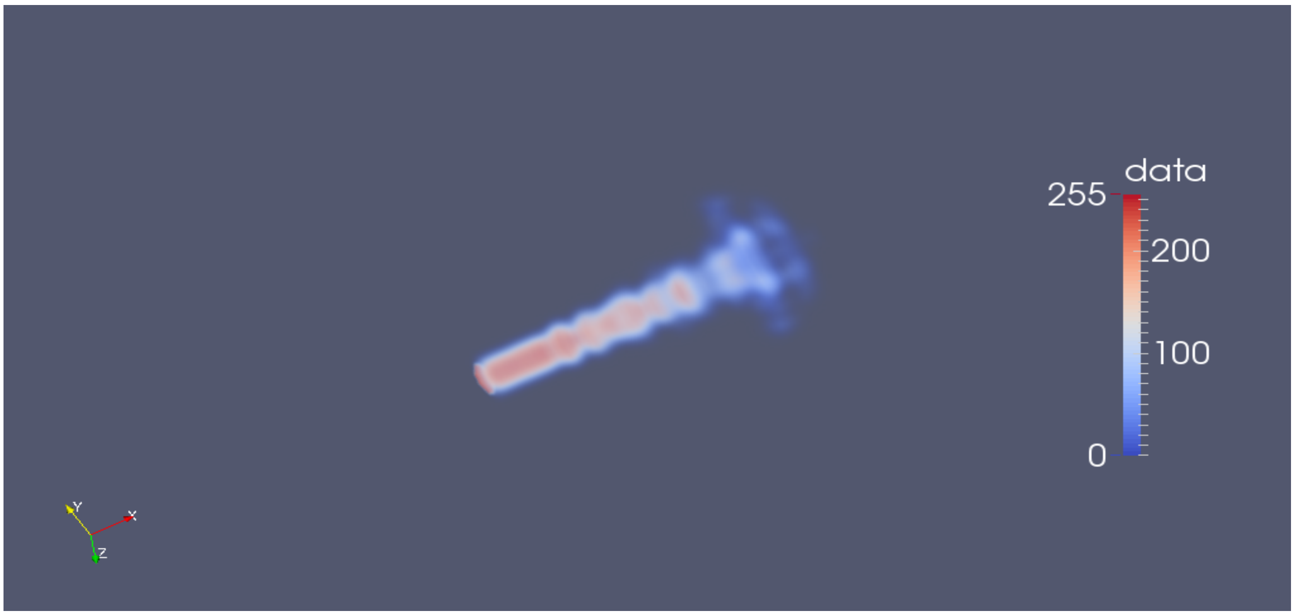
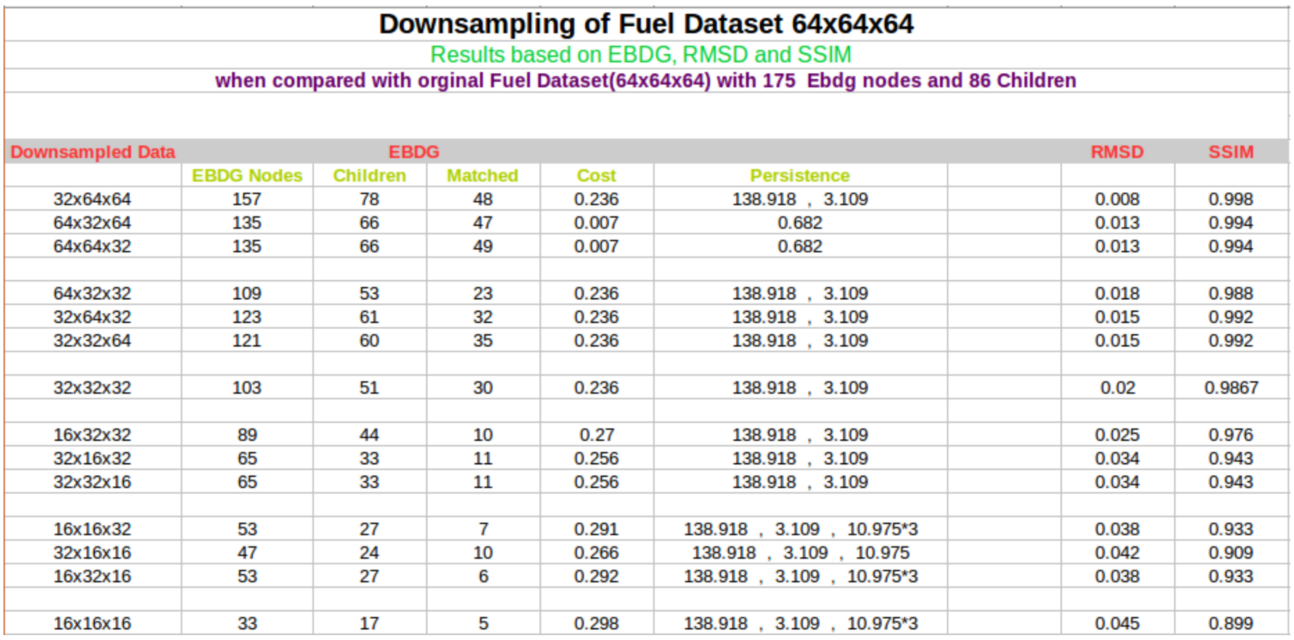


Figure 6.1: Experiments performed on Fuel dataset, resolution $64 \times 64 \times 64$, storing 8 bits per voxel. Shows simulation of fuel injection into a combustion chamber

The table in **Figure 6.2** shows the results for three measures considered that are eBDG based cost, RMSD and SSIM.



Downsampling of Fuel Dataset 64x64x64

Results based on EBDG, RMSD and SSIM

when compared with orginal Fuel Dataset(64x64x64) with 175 Ebdg nodes and 86 Children

Downsampled Data	EBDG						RMSD	SSIM
	EBDG Nodes	Children	Matched	Cost	Persistence			
32x64x64	157	78	48	0.236	138.918 , 3.109		0.008	0.998
64x32x64	135	66	47	0.007	0.682		0.013	0.994
64x64x32	135	66	49	0.007	0.682		0.013	0.994
64x32x32	109	53	23	0.236	138.918 , 3.109		0.018	0.988
32x64x32	123	61	32	0.236	138.918 , 3.109		0.015	0.992
32x32x64	121	60	35	0.236	138.918 , 3.109		0.015	0.992
32x32x32	103	51	30	0.236	138.918 , 3.109		0.02	0.9867
16x32x32	89	44	10	0.27	138.918 , 3.109		0.025	0.976
32x16x32	65	33	11	0.256	138.918 , 3.109		0.034	0.943
32x32x16	65	33	11	0.256	138.918 , 3.109		0.034	0.943
16x16x32	53	27	7	0.291	138.918 , 3.109 , 10.975*3		0.038	0.933
32x16x16	47	24	10	0.266	138.918 , 3.109 , 10.975		0.042	0.909
16x32x16	53	27	6	0.292	138.918 , 3.109 , 10.975*3		0.038	0.933
16x16x16	33	17	5	0.298	138.918 , 3.109 , 10.975*3		0.045	0.899

Figure 6.2: Downsampling of Fuel Dataset

The results for various downsampled versions are computed when compared with original fuel dataset i.e. $64 \times 64 \times 64$.
The RMSD value **0** corresponds to identical datasets and greater than **0** and less than **1** implies different datasets. More the value of RMSD means more structural difference. SSIM value of **1** implies structurally identical datasets. And as values goes towards **0**, implies more difference.
The first column in the table mentions the dimensions of various downsampled versions of fuel dataset. Under eBDG Column, in the first column it shows the total number of eBDG nodes, where each node refers to a unique sub tree in the join tree. And we are interested only in the last node (corresponds to whole join tree). The second column mentions the number of children of this special last node. And then in next column it shows the number of matched eBDG nodes. After that the table shows the eBDG cost. And next to it are the prominent persistence values of the unmatched eBDG nodes.
Next, the table has RMSD and SSIM values.

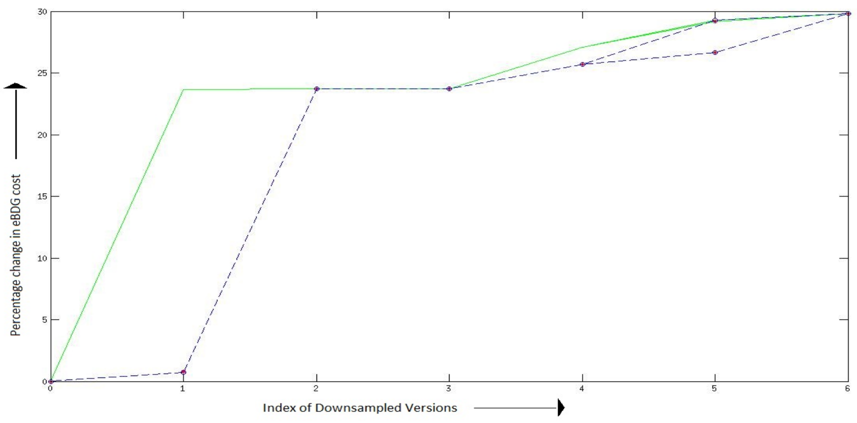
The table on a whole gives an idea for various downsampled versions as how the topology and structure of datasets change when we downsample the original scalar field.

The plots in **Figures 6.3, 6.4, and 6.5** show the percentage change based on eBDG, RMSD and SSIM measures considered respectively, when we keep on downsampling the dataset starting from 64 × 64 × 64 till we reach 16 × 16 × 16 resolution in Fuel.

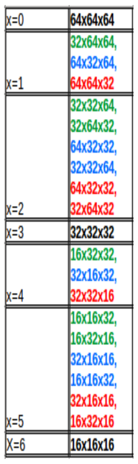
The green line plot is for x-dimension, blue is for y-dimension and red is for z-dimension being downsampled gradually in above mentioned dimensions respectively.

On the x- axis of the plots, the cases are numbered as mentioned below:

1. x=0 for 64 × 64 × 64 fuel.
2. x=1 for 32 × 64 × 64 for green line, 64 × 32 × 64 for blue and 64 × 64 × 32 for red.
3. x=2 are for 32 × 32 × 64 and 32 × 64 × 32 for green and 64 × 32 × 32 and 32 × 32 × 64 for blue and 64 × 32 × 32 and 32 × 64 × 32 for red.
4. x=3 is for 32 × 32 × 32 fuel.
5. x=4 for 16 × 32 × 32 for green line, 32 × 16 × 32 for blue and 32 × 32 × 16 for red.
6. x=5 are for 16 × 16 × 32 and 16 × 32 × 16 for green and 32 × 16 × 16 and 16 × 16 × 32 for blue and 32 × 16 × 16 and 16 × 32 × 16 for red.
7. x=6 is for 16 × 16 × 16 fuel.



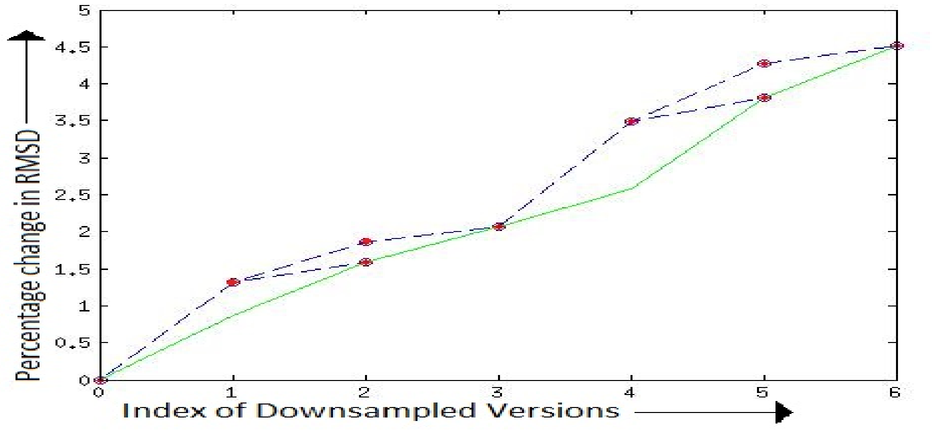
(a) Plot shows the percentage change of eBDG cost during downsampling. The indices on x-axis correspond to the downsampled versions of fuel dataset as mentioned earlier in Results section and y-axis shows the percentage of eBDG cost.

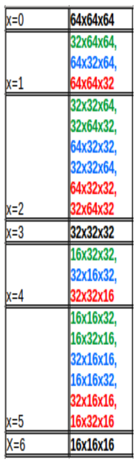


x=0	64x64x64
x=1	32x64x64, 64x32x64, 64x64x32
x=2	32x32x64, 32x64x32, 64x32x32, 32x32x64, 64x32x32, 32x64x32
x=3	32x32x32
x=4	16x32x32, 32x16x32, 32x32x16
x=5	16x16x32, 16x32x16, 32x16x16, 16x16x32, 32x16x16, 16x32x16
X=6	16x16x16

(b) Downsampled Versions on X-Axis

Figure 6.3: eBDG Cost Plot



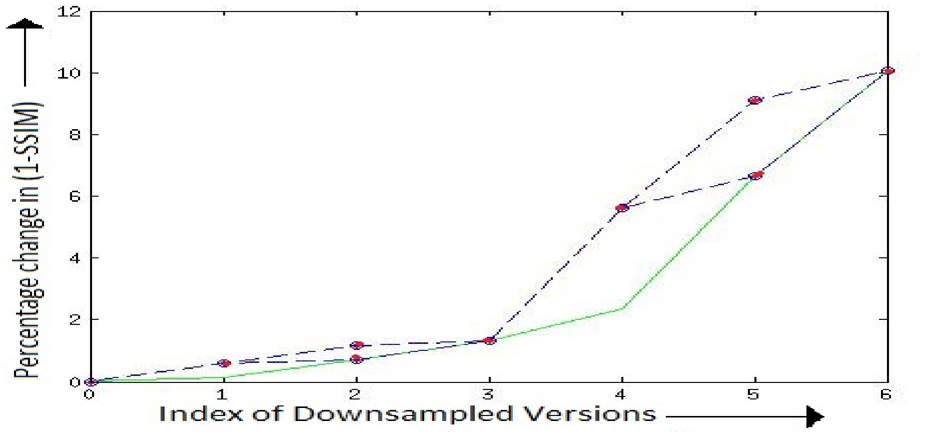


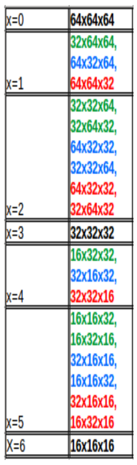
x=0	64x64x64
x=1	32x64x64, 64x32x64, 64x64x32
x=2	32x32x64, 32x64x32, 64x32x32, 32x32x64, 64x32x32, 32x64x32
x=3	32x32x32
x=4	16x32x32, 32x16x32, 32x32x16
x=5	16x16x32, 16x32x16, 32x16x16, 16x16x32, 32x16x16, 16x32x16
X=6	16x16x16

(a) Plot shows the percentage change of RMSD values during downsampling. The indices on x-axis correspond to the downsampled versions of fuel dataset as mentioned earlier in Results section and y-axis shows the percentage change of RMSD values.

(b) Downsampled Versions on X-Axis

Figure 6.4: RMSD Plot





x=0	64x64x64
x=1	32x64x64, 64x32x64, 64x64x32
x=2	32x32x64, 32x64x32, 64x32x32, 32x32x64, 64x32x32, 32x64x32
x=3	32x32x32
x=4	16x32x32, 32x16x32, 32x32x16
x=5	16x16x32, 16x32x16, 32x16x16, 16x16x32, 32x16x16, 16x32x16
X=6	16x16x16

(a) Plot showing the percentage change of (1-SSIM) values during downsampling. The indices on x-axis correspond to the downsampled versions of fuel dataset as mentioned earlier in Results section and y-axis shows the percentage of (1-SSIM) values.

(b) Downsampled Versions on X-Axis

Figure 6.5: SSIM Plot

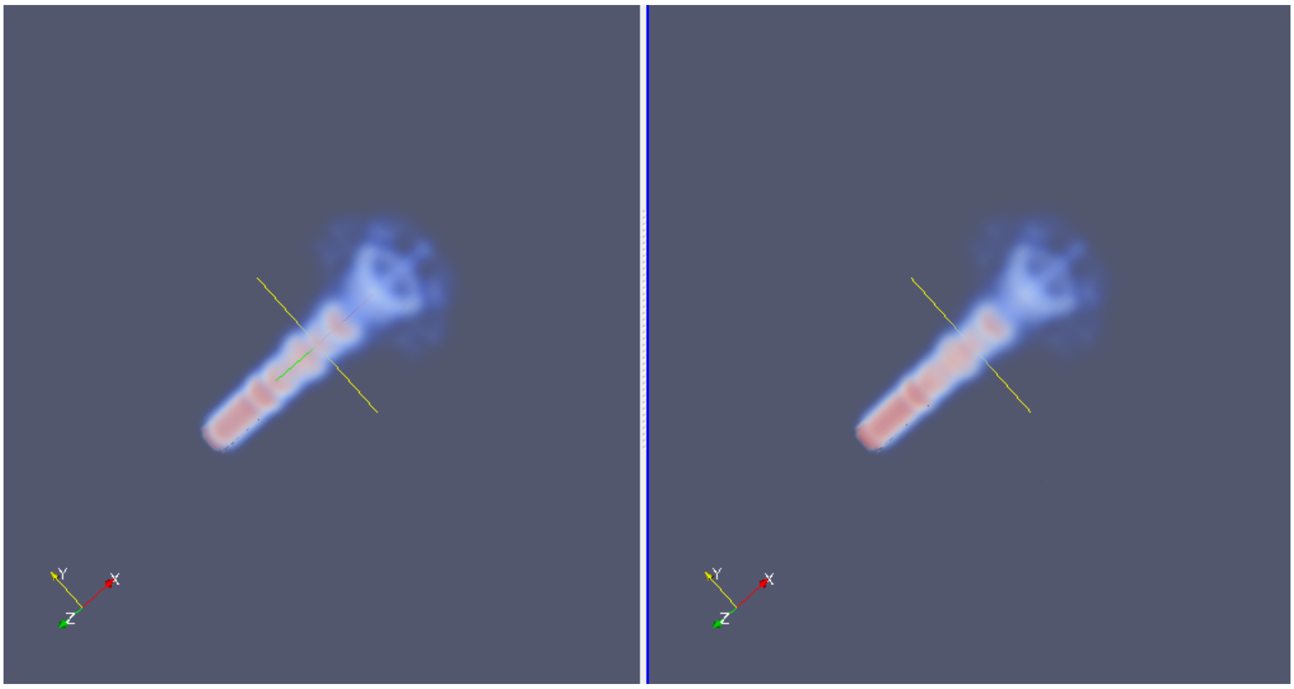


Figure 6.6: Left image shows $64 \times 64 \times 64$ Fuel dataset and the right image shows the down-sampled $32 \times 64 \times 64$ Fuel dataset.

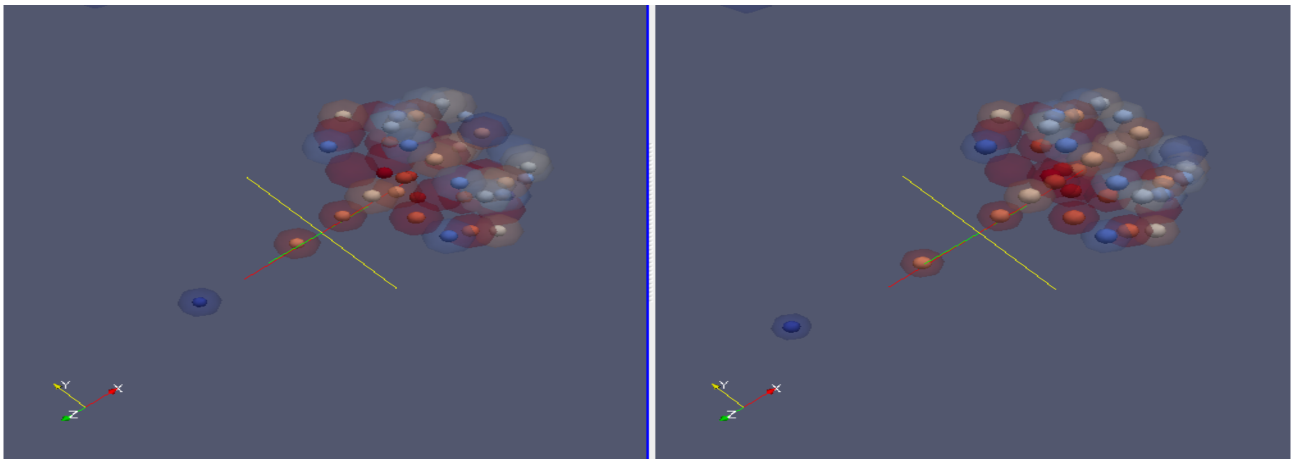


Figure 6.7: Left and right image shows the matchings obtained based on only persistence and persistence and position considered together in $64 \times 64 \times 64$ (left) when compared with $32 \times 64 \times 64$ (right) . For both images the larger spheres (less opaque ones) correspond to matches based on persistence. The smaller ones, more opaque spheres correspond for matches when both persistence and position are taken into account.

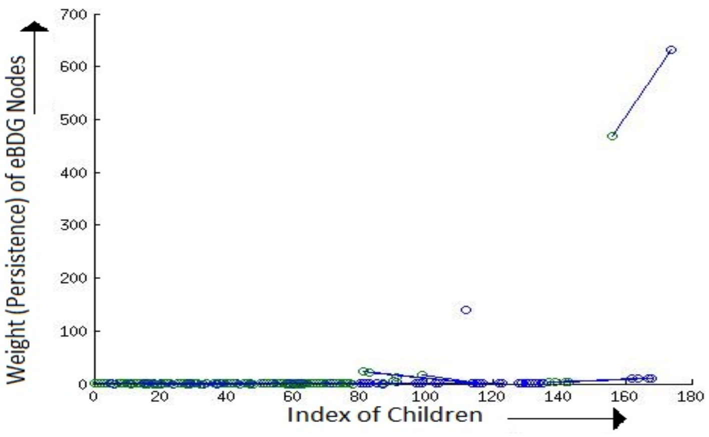


Figure 6.8: Children, of eBDG node that maps to the complete join tree of original $64 \times 64 \times 64$ fuel dataset and eBDG node from eBDG of $32 \times 64 \times 64$ downsampled dataset that has minimum eBDG score, are plotted in blue and green respectively. The lines are drawn between the ones matched.

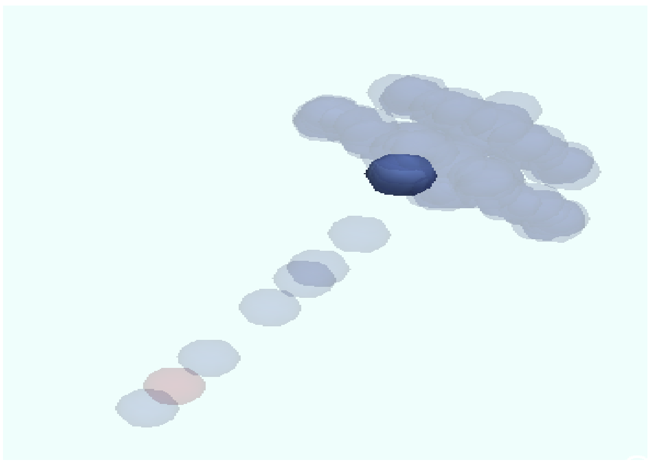


Figure 6.9: The image shows the critical points corresponding to the children nodes of eBDG node that maps to the complete join tree of original $64 \times 64 \times 64$ fuel dataset. The most opaque sphere corresponds to the eBDG node with highest persistence that did not match.

Chapter 7

Observations

Observations based on the plots shown in **Figures 6.3, 6.4, 6.5** :

1. In **Figure** [**6.3**], the eBDG cost can be seen to substantially increase when we drop points in x-dimension. This is justified because in Fuel dataset, the data is concentrated along x-axis.

2. In **Figure** [**6.3**], at index 1 on the x-axis of the plot (for $32 \times 64 \times 64$ resolution), there is sudden increase in topology change. This is due to a very high persistent feature (eBDG weight 138.918) being lost on dropping points in x-dimension. While for $64 \times 32 \times 64$ and $64 \times 64 \times 32$ at index 2 on the x-axis of the same plot, the percentage change is extremely less. The same can be verified from the details given under persistence column in **Figure** [**6.2**]. Further downsampling results in addtional loss of topological features but all with less persistence.

3. While in RMSD plot and SSIM plot (**Figure** [**6.4**], **Figure** [**6.5**] respectively), dropping points in y and z dimension overshadows the cost of dropping points in x-dimension.

In **Figure** [**6.6**], the original fuel dataset i.e. with resolution $64 \times 64 \times 64$ and the downsampled version with $32 \times 64 \times 64$ resolution are shown in paraview. For the same datasets **Figure** [**6.7**] shows the matching obtained by eBDG measure, based on only persistence and persistence and position considered together.

For these we can observe:

1. The cardinality of matches obtained based on persistence and position together is less than the cardinality of matches obtained by considering only persistence. This can be

justified because an additional constraint of position is included along with persistence which leads to less number of matches.

2. Few matches out of the ones obtained by considering both persistence and position are completely different from the ones obtained by considering only persistence. The same can be verified by seeing exclusively placed solid small spheres. This can be explained because it is more likely that the critical point has moved to a point in its neighborhood during downsampling and should be matched with a critical point nearby rather than matching it based on persistence alone. So considering position and persistence together can result in few matches that are different from matches obtained by considering only persistence.

Figure [**6.8**] shows the matchings between the children of eBDG node, from eBDG of original $64 \times 64 \times 64$ fuel dataset, that maps to complete join tree and eBDG node from eBDG of $32 \times 64 \times 64$ downsampled dataset that has minimum eBDG score. The plot clearly shows that there is one high persistent eBDG node (blue point located approximately at center) which is unmatched and rest unmatched nodes are with very low persistence. This unmatched high persistent node can be visualized in paraview as shown in **Figure** [**6.9**].

Chapter 8

Conclusions and Future Work

In the study we have considered one downsampling algorithm to see how the process of downsampling results in topology change. We considered eBDG measure to quantify the topology change and compared with RMSD and SSIM values. The matchings obtained by eBDG measure can be visualized and analysed in paraview.

In nutshell, considering these three measures together helps to compare topology and structural changes that results from downsampling. One can decide on sampling techniques to be considered that result in less topology change by analysing the costs. As the results help to visualize and quantify topology changes during downsampling, it can help to guide where one should stop during downsampling.

The work can be extended for other approaches where we require to compare scalar fields defined on the same domain like studying time varying scalar fields.
This work can also contribute in the field of comparing and analysing multifield data.

Bibliography

[1] Aditya Acharya and Vijay Natarajan. A parallel and memory efficient algorithm for constructing the contour tree. In *2015 IEEE Pacific Visualization Symposium (PacificVis)*, pages 271–278. IEEE, 2015.

[2] Pankaj K Agarwal, Lars Arge, Thomas Mølhave, Morten Revsbæk, and Jungwoo Yang. Maintaining contour trees of dynamic terrains. *arXiv preprint arXiv:1406.4005*, 2014.

[3] Chandrajit L Bajaj and Daniel R Schikore. Topology preserving data simplification with error bounds. *Computers & Graphics*, 22(1):3–12, 1998.

[4] Roger L Boyell and Henry Ruston. Hybrid techniques for real-time radar simulation. In *Proceedings of the November 12-14, 1963, fall joint computer conference*, pages 445–458. ACM, 1963.

[5] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

[6] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction.* American Mathematical Soc., 2010.

[7] H Freeman and SP Morse. On searching a contour map for a given terrain elevation profile. *Journal of the Franklin Institute*, 284(1):1–25, 1967.

[8] Thomas Gerstner and Renato Pajarola. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. In *Proceedings of the conference on Visualization'00*, pages 259–266. IEEE Computer Society Press, 2000.

[9] Christopher Gold and Sean Cormack. Spatially ordered networks and topographic reconstructions. *International Journal of Geographical Information System*, 1(2):137–148, 1987.

[10] Martin Kraus and Thomas Ertl. Topology-guided downsampling. In *Volume Graphics 2001*, pages 223–234. Springer, 2001.

[11] In So Kweon and Takeo Kanade. Extracting topographic terrain features from elevation maps. *CVGIP: image understanding*, 59(2):171–182, 1994.

[12] Dmitriy Morozov, Kenes Beketayev, and Gunther Weber. Interleaving distance between merge trees. *Discrete and Computational Geometry*, 49:22–45, 2013.

[13] Vijay Natarajan and Valerio Pascucci. Volumetric data analysis using morse-smale complexes. In *International Conference on Shape Modeling and Applications 2005 (SMI'05)*, pages 320–325. IEEE, 2005.

[14] Valerio Pascucci, Kree Cole-McLaughlin, and Giorgio Scorzelli. Multi-resolution computation and presentation of contour trees. In *Proc. IASTED Conference on Visualization, Imaging, and Image Processing*, pages 452–290. Citeseer, 2004.

[15] Himangshu Saikia, Hans-Peter Seidel, and Tino Weinkauf. Extended branch decomposition graphs: Structural comparison of scalar data. In *Computer Graphics Forum*, volume 33, pages 41–50. Wiley Online Library, 2014.

[16] Jayanta K Sircar and Juan A Cebrian. Application of image processing techniques to the automated labelling of raster digitized contour maps. In *Proceedings of the 2nd International ACM Symposium on Spatial Data Handling*, pages 171–184, 1986.

[17] Bong-Soo Sohn. Topology preserving tetrahedral decomposition of trilinear cell. In *International Conference on Computational Science*, pages 350–357. Springer, 2007.

[18] Shigeo Takahashi, Tetsuya Ikeda, Yoshihisa Shinagawa, Tosiyasu L Kunii, and Minoru Ueda. Algorithms for extracting correct critical points and constructing topological graphs from discrete geographical elevation data. In *Computer Graphics Forum*, volume 14, pages 181–192. Wiley Online Library, 1995.

[19] Shigeo Takahashi, Yuriko Takeshima, and Issei Fujishiro. Topological volume skeletonization and its application to transfer function design. *Graphical Models*, 66(1):24–49, 2004.

[20] Debasish Tapna. Interactive volume rendering of large scalar fields on mobile devices using subsampling techniques. Master's thesis, Department of Computer Science and Automation, Indian Institute of Science, Bangalore, 6 2011.

[21] Dilip Mathew Thomas and Vijay Natarajan. Symmetry in scalar field topology. *IEEE transactions on visualization and computer graphics*, 17(12):2035–2044, 2011.

[22] Marc Van Kreveld, René van Oostrum, Chandrajit Bajaj, Valerio Pascucci, and Dan Schikore. Contour trees and small seed sets for isosurface traversal. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 212–220. ACM, 1997.

[23] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[24] Gunther H Weber, Scott E Dillard, Hamish Carr, Valerio Pascucci, and Bernd Hamann. Topology-controlled volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):330–341, 2007.