Query-Driven Analysis and Visualization for Large-Scale Scientific Dataset using Geometry Summarization and Bitmap Indexing

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Tzu-Hsuan Wei, B.S., M.S.

Graduate Program in Computer Science and Engineering

The Ohio State University

2017

Dissertation Committee:

Han-Wei Shen, Advisor

Gagan Agrawal

Yusu Wang

Abstract

The computational power of modern supercomputers grows rapidly, and it facilitates scientists to produce high-resolution datasets when simulating physical or weather models, which generate extreme scale data with multiple variables most of the time. However, storage, transmission, or exploration of such large-scale data is challenging. In the past decades, several visualization approaches have been developed to effectively explore datasets by displaying underlying information of datasets. Query-driven visualization is one of the prominent approaches, as it significantly reduces visual exploration time by only focusing on interesting or important features for further analysis and decision making. However, as the size of scientific datasets becomes too large, traditional data exploration approaches become ineffective. An emerging approach is to create *data summarizations* to first reduce the size of the dataset, and then perform data exploration on the data summarization. An ideal data summarization aims at preserving the characteristics of the raw data as much as possible while keeping the size small. However, to retrieve salient features from the raw data and create such importance-based data summarizations is challenging. In this dissertation, we address the issues that need to be solved when applying query-driven analysis and visualization using data summarizations.

First, we focus on the problem of identifying salient features and evaluating selected features for creating a data summarization. To analyze a volumetric dataset, displaying isosurface is typically used to reveal the locations of values that the user focuses on. We

propose a novel algorithm to select salient features described by isosurfaces and evaluate the chosen isosurfaces quantitatively. Our approach applies information theory to examine how much information left in an enclosed volume between two selected isosurfaces and then determine whether any isosurfaces should be included to enhance the information of surface summaries. Second, we focus on the task to efficiently search for the user-queried features using existing data summarization. We propose a novel algorithm to detect local histogram-based features with high performance by leveraging bitmap index. Rather than exhaustively searching for the target local histograms at all data voxels, our approach quickly determines the search space which is much less than all data points and detect the voxels whose local histograms match the user-defined histogram. Furthermore, we also propose two algorithms to solve the performance issues when extending the approach of efficient histogram-based feature search to multi-field datasets. Third, we aim to improve an existing data summarization in order to provide a better one in terms of the quality of the content and the size of the data. We propose two approaches to tackle the limitations of the bitmap data summarization. The first one is an adaptive sampling approach using bitmap index called information guided stratified sampling (IGStS) for creating a sampling-based bitmap that preserves the important characteristics of the raw data. Furthermore, we propose a novel data recovery approach to reconstruct the irregular subsampled dataset into a volume dataset with regular grid structure for qualitative post-hoc data exploration and visualization. We transform the data recovery problem to an optimal assignment problem and solve the value assignment problem by the Hungarian algorithm. The quantitative and visual efficacy of our proposed data sampling and recovery approaches are demonstrated through multiple experiments and applications.

Acknowledgments

First and foremost, I would like to express my deep gratitude to my advisor Dr. Han-Wei Shen, for his support, patience, inspiring discussions and professional guidance throughout my time at The Ohio State University. His training on logical thinking has influenced and changed me greatly. Without his guidance and encouragement, this doctoral dissertation would not have been possible to complete.

I would also like to thank Dr. Gagan Agrawal and Dr. Yusu Wang for serving on my dissertation committee and providing insightful advice.

Sincerely thanks to Dr. James Ahrens and Dr. Jon Woodring from the Los Alamos National Laboratory, for mentoring and guiding me in relevant projects.

Many thanks to the past and present members of the Graphics and Visualization Study (GRAVITY) research group. During my Ph.D. research in the group, I learned many professional skills from senior students and visiting scholars including Teng-Yok Lee, Abon Chaudhuri, Chun-Ming Chen, and Huijie Zhang. I also would like to thank to my lab colleagues, Steve Martin, Boonthanome Nouanesengsy, Xiaotong Liu, Ayan Biswas, Xin Tong, Kewei Lu, Soumya Dutta, Ko-Chih Wang, Cheng Li, Wenbin He, Subhashis Hazarika, and Junpeng Wang. I enjoy the time working with them.

Finally, I would like to thank the ones that are most important to me: my parents, Chin-Feng Wei, Mei-O Wei Chien, and my wife, Yi-Jie Kao. Without their support, encouragement, and love, I would not be able to complete my Ph.D. study.

Vita

December 5, 1980 ..... Born - Taipei, Taiwan

2003 ..... B.S. Atmospheric Sciences

2005 ..... M.S. Computer Science and Information Engineering

2011-present ..... Graduate Research Associate, The Ohio State University.

2013-2016 ..... Graduate Teaching Associate, The Ohio State University.

Jun-August, 2016 ..... Research Intern, Los Alamos National Lab.

Publications

Research Publications

Tzu-Hsuan Wei, Chun-Ming Chen, Jonathan Woodring, Huijie Zhang and Han-Wei Shen "Efficient Distribution-based Feature Search in Multi-field Datasets". *IEEE Pacific Visualization Symposium, PacificVis 2017*, 121–130, April 2017.

Ko-Chih Wang, Kewei Lu, Tzu-Hsuan Wei, Naeem Shareef, Han-Wei Shen "Statistical Visualization and Analysis of Large Data Using a Value-based Spatial Distribution". *IEEE Pacific Visualization Symposium, PacificVis 2017*, 121–130, April 2017.

Tzu-Hsuan Wei, Chun-Ming Chen, and Ayan Biswas "Efficient Local Histogram Searching via Bitmap Indexing". *EuroVis 2015, Computer Graphics Forum*, 34(3):81–90, May 2015.

Abon Chaudhuri, Tzu-Hsuan Wei, Teng-Yok Lee, Han-Wei Shen and Tom Peterka "Efficient Range Distribution Query for Visualizing Scientific Data". *IEEE Pacific Visualization Symposium, PacificVis 2014*, 201–208, March 2014.

Tzu-Hsuan Wei, Teng-Yok Lee, and Han-Wei Shen "Evaluating Isosurfaces with Level-set-based Information Maps". *EuroVis 2013, Computer Graphics Forum*, 32(3):1–10, May 2013.

Fields of Study

Major Field: Computer Science and Engineering

Studies in:

Scientific Visualization
Computer Graphics
High-Performance Computing

Table of Contents

List of Tables

List of Figures

Figure **Page**

Chapter 1: Introduction

1.1 Background and Motivation

The computation power of modern supercomputers grows rapidly in recent years. According to the latest report of the TOP500 supercomputers released in June 2016, the most powerful and fastest supercomputer, Sunway TaihuLight, is able to reach the peak performance more than 125 peta-FLOPS (125×10^{15} floating-point operations per second) which is about 2.28 times faster than the previous holder of record, Tianhe-2, built up in 2013. Benefiting from the huge computation power, scientists are able to run high-resolution scientific simulations in various scientific disciplines, such as climate modeling and computational fluid dynamic, and produce large-scale datasets in the order of petabytes and beyond. To store, manage, and explore such large-scale scientific datasets, appropriate approaches are crucial.

In the past decades, several visualization approaches have been developed to effectively explore datasets by displaying underlying information of datasets. Query-driven visualization is one of the prominent approaches, as it significantly reduces visual exploration time and allows users (scientists) to focus only on a small part of interesting data for further analysis and decision making. However, as the size of scientific datasets becomes large, traditional data exploration approaches become ineffective. An emerging approach is to

create a data summarization to first reduce the size of the dataset, and then apply data processing and analysis on the data summarization. An ideal data summarization aims at preserving the characteristics of the raw data as much as possible while keeping the size small. However, to retrieve salient information from the raw data and create such data summarization is not a trivial task. The challenges are listed below:

- In order to efficiently explore a large-scale dataset, scientists usually only retrieve interesting or important features in the dataset for analysis rather than navigating the entire dataset. However, it is a crucial problem to determine salient features automatically from the dataset. Furthermore, when particular features are selected for summarizing salient information in the data, it is important to know how much information is in the selected features and what meaningful features are missing and necessary to be included. Therefore, an approach of selecting salient features as well as a quantitative evaluation of the selected features are required.

- Besides determining important features automatically from the data, domain knowledge is usually introduced to improve the performance and the quality of data exploration. During the process of analysis, scientists first determine interesting or important features that they desire to focus on and then look for other regions that have similar features in the whole data domain. However, it is a challenging task to quickly discover features in the entire data domain, especially in a large-scale dataset. Therefore, an efficient approach to search for the user-queried features is required.

- Scientific datasets are mostly associated with multiple fields, such as pressure and temperature fields in climate datasets. There is a need to define feature descriptors with combined fields since some features cannot be extracted just from a single field.

However, exploring high-dimensional features in a multi-field dataset is challenging due to high computational cost. An efficient approach to overcome the curse of dimensionality for feature detection in a multi-field dataset is thus required.

- A bitmap index is an efficient indexing structure that has been applied in many applications of query-driven visualization. Because of the advantage of bitwise operations supported by the computer hardware, it is able to quickly respond to the data value query. In addition to providing efficient query responses, bitmap indexing can also be applied to reduce the data size overhead by compression, which can be regarded as a data summarization. In general, the size of a bitmap after compression can be less than 30% in size compared with the raw data. However, it could be still too large to be stored or transferred when the size of the original raw data is extremely large. Therefore, an effective approach that allows further reduction of the size of compressed bitmaps while keeping the characteristics of the raw data as much as possible is thus required.

In this dissertation, we propose several novel approaches to tackle the above challenges. The first part of the dissertation focuses on developing techniques for feature-based analysis and visualization using data summarizations. The second part of the dissertation focuses on improving an existing data summarization in order to provide a summarization that can preserve the essential characteristics of the raw data while keeping the size small.

1.2 Feature-based Analysis and Visualization using Data Summarizations

1.2.1 Defining and Evaluating Representative Isosurfaces for the Large-Scale Data Analysis and Visualization

Several feature descriptors have been applied for exploring large-scale scalar fields in the past two decades. Isosurface is one of the popular feature descriptors which captures the characteristics of a scalar value across the data space. Given a value, it is a geometric surface that only contains points having the same value, which is extracted by surface extraction techniques such as the Marching Cubes algorithm. To reduce the data exploration time, a common method is to select a set of isosurfaces as a geometry summarization for efficient scalar field analysis and visualization. Several approaches have been developed to guide the selection of salient isosurfaces from an infinite number of isosurfaces. These approaches exploit the statistical features of dataset [4, 14, 28, 75], geometric features [46], topological features [15, 16, 101], as well as features that can be derived from distance transforms [13]. While several approaches proposed for selecting salient isosurfaces based on certain criteria, few efforts are put on quantitatively evaluating the representativeness of the selected isosurfaces for the entire scalar field. Without evaluations, it is difficult to know whether the pre-selected isosurfaces capture all important features, or if any spatial subregions or value intervals are under-represented.

In Chapter 3, we present a framework for selecting and evaluating the isosurfaces based on the information theory. In this research, we assume that if there are no unexpected isosurfaces within an enclosed subvolume between two isosurfaces, these two isosurfaces are considered representative for this subvolume. On the contrary, if any surprising isosurfaces exist in the enclosed subvolume, adding new isosurfaces is necessary in order to reduce

the unknown information in the subvolume. To discover unexpected isosurfaces within an enclosed subvolume between two isosurfaces, we examine each intermediate surface generated by morphing from one isosurface to the other using the level set method [12]. From these morphed surfaces, we determine if they are representative for any true isosurfaces in the subvolume by sampling on the surface. For each morphed surface, if all samples on the surface have the same value, this surface is identical to one of the true isosurfaces. On the contrary, if the sample values exhibit a larger variance, i.e., a higher entropy, then this surface is not aligned well with any true isosurfaces in the field. When a majority of the morphed surfaces in the region are not aligned well with the true isosurfaces, we consider that some unexpected isosurfaces exist in the subvolume so that more isosurfaces are necessarily added to cover missing information. Based on the quantitative evaluation approach, the proposed algorithm for automatic selection of isosurfaces facilitates more effective scalar field analysis.

1.2.2 Efficient Local Histogram-based Feature Search via Bitmap Data Summarization

Representing features by local histograms is another proven technique in several volume analysis and visualization applications. The local histogram at a voxel is constructed by collecting data values in the neighborhood around that voxel, which describes the dispersion of values in that local region. It has been shown that histograms in local regions can be used to identify material boundaries [85], predict material composition in a local area[52] and improve the transfer function design[58]. The computational performance of local histogram search, however, has not drawn much attention in the past works. For the traditional approach of local histogram search, we need to compute a histogram from each local region and compare it with the pre-defined target histogram. This time-consuming

exhaustive search prohibits the aforementioned applications from being practical in large data cases.

In Chapter 4, we present an efficient algorithm to locate voxels whose local histograms in their neighborhood match the user-defined target histogram. It is achieved by leveraging bitmap index which is designed to efficiently locate data that matches queried values and have been adopted in several applications. In this research, we observe the value range defined in the user-defined target histogram is usually a subset of the entire data value range. Based on the observation, we propose an efficient local histogram search algorithm that only scans through the voxels within the value range of user-defined histograms and their neighborhood rather than scanning through all data voxels. The search region can be reduced from the whole data space to just a portion of it. Compared to the conventional approaches, the proposed approach can search for the user-defined target feature more efficiently.

1.2.3 Efficient Distribution-based Feature Search in Multi-field Datasets

In the past works, the feature-based data analysis and visualization mostly focused on the single-field dataset and disregarded the interrelation information between different fields. However, more and more scientific problems cannot only be solved by analyzing only one single field; some phenomena can be only captured from a combined of multiple fields. Therefore, defining features with combined fields is necessary for qualitative data analysis and visualization. To achieve it, we extend the idea of histogram-based feature search introduced in Chapter 4 to the multi-field domain.

In Chapter 5, we present an efficient local distribution search for two types of multi-field features - marginal and joint features. For the marginal feature search, the target distribution is defined accordingly in each field, and the features are searched for independently. For the joint feature search, the target distribution is defined by a joint distribution which can be extracted from a joint set of fields. We then search for the joint features by comparing local joint distributions to the target joint distribution. Experiments show that our local distribution-based feature search for multi-field datasets performs much faster than the previous methods, and the total run-time memory usage remains small when the number of fields increases.

1.3 An Improved Data Summarization for Large-Scale Scalar Data Analysis and Visualization

1.3.1 Information Guided Data Sampling and Recovery using Bitmap Index

In Chapter 4 and 5, we utilize compressed bitmap index to achieve efficient histogram-based feature detection. However, we observe the effectiveness of existing compression scheme for the bitmap index is limited. In most of the cases, although the size of compressed bitmaps can be less than 30% in size compared with the raw data [82], it can be still too large to be stored or transferred. An existing solution [81] is to perform sampling on a generated bitmap, which can further reduce the size of the bitmap by improving the effectiveness of run-length compression. However, existing approaches do not take the data complexity into account when performing sampling on the generated bitmap, which can lead to loss of information in the region with high data complexity but keep redundant information in the region with low data complexity. Therefore, there is a need to sample

data adaptively to present more representative samples in the subsampled data while keeping the size small. Furthermore, since the bitmap index incorporating with data sampling is usually stored as an irregular grid volume, there is a need to recover samples from the irregular grid data into a regular grid volume when performing specific data visualization techniques, such as volume rendering or isosurface rendering.

In Chapter 6, we propose two approaches to solve the issues mentioned above. First, we propose an information guided data sampling approach applied on bitmap index. We apply stratified sampling that subdivides the raw samples into several groups and draws samples from each group. Instead of drawing the same percentage of samples from each group, we determine the number of samples drawn from each group of data by evaluating its data complexity according to the entropy value. Through this adaptive sampling approach, the proposed bitmap data summarization enables to provide superior quality of data analysis and visualization while keeping the size small. Second, we propose a novel data recovery approach for recovering the data samples from the bitmap data summarization incorporating with data sampling. We transform the data recovery problem to an optimal assignment problem and solve the value assignment problem by the Hungarian algorithm [51]. The quantitative and visual efficacy of our proposed data sampling and recovery approaches are demonstrated through multiple experiments and applications.

Thesis Organization The rest of the proposal is organized as follows: Chapter 2 provides the background techniques of our work and lists recent researches on related topics. Chapter 3 elaborates the proposed approach of selecting and evaluating isosurfaces. Chapter 4 explains the efficient approach of local histogram-based feature search. Chapter 5 describes our work on extending the histogram-based feature search to multi-field datasets. Chapter

6 introduce our improved bitmap data summarization. Finally, Chapter 7 summarizes the works and outlines the future research plans.

Chapter 2: Background and Relate Works

2.1 Data Analysis and Visualization via Isosurfacce

Researchers have studied various issues related to the analysis of scalar fields via isosurfaces. For example, the areas of isosurfaces can be used to construct the distribution of scalar values in the entire field [4, 14, 75]. The topology of isosurfaces and its derived data structures such as contour trees or Reeb graphs can provide an overview of the scalar field structure [15, 16, 101]. To detect the isosurfaces as the material boundaries, local statistics and histograms are also utilized [68, 84]. Recently, certain geometric properties of isosurfaces such as fractal dimensions [46] and distance transforms [13] are utilized to measure the salience of isosurfaces. However, to our knowledge, not much attention has been put on evaluating how a given set of isosurfaces selected for visualization can represent the entire scalar field, and which spatial regions or value intervals are still under-represented, so the visualization needs to be refined.

2.2 Information Theory for Data Visualization

Information theory [26] has been extensively applied to enhance the visual quality of data visualization. Gumhold [33] presented a tool to automatically place light sources for given camera parameters, which results in maximal information content on the scene

through the illumination. Vázquez et al. [86] introduced the viewpoint entropy for automatically selecting good viewpoints that cover all visible polygons on the scene. The approach can minimize the storage of representative images with good views in the image-based representation. Bordoloi and Shen [7] proposed a view selection method designed for volume rendering by measuring the ”goodness” of each viewpoint using the voxel-based entropy value. Viola et al. [87] proposed an approach to automatically determine the optimal viewpoints when visualizing the user-defined features through maximizing the mutual information between a set of viewpoints and the features in the dataset. Bramon et al. [10] presented a mutual information-based method to evaluate the information transfer between the source dataset and the rendered image, which solves three fundamental visualization problems: selection of viewpoints, transfer function design, and light positioning.

Information theory has been applied to enhance the visual quality of flow visualization as well. Xu et al. [100] introduced an information-theoretic model to quantitatively measure the complexity in the flow field and place streamlines in the regions with high complexity to enhance the quality of the streamline visualization. Lee et al. [53] introduced an approach called maximal entropy projection for streamline filtering and viewpoint selection, which is applied to suggest the best viewpoint and increase the visibility of the salient flow features. Tao et al. [83] proposed an information-theoretical approach to examining the representativeness of streamlines and viewpoints, which is applied to define a camera path that passes through the important viewpoints for automatic flow data exploration.

In addition, information theory has been applied to explore time-varying, multimodal or multivariate dataset. Wang et al. [89] introduced a measurement of importance to select salient time steps that contain the maximal amount of information in a volumetric time-varying data. Bramon et al. [9] proposed a mutual information-based approach to provide

informative multimodal visualization by fusing the most relevant information from different medical datasets. Biswas et al. [6] employed the surprise and predictability metrics between variables, which is similar to the concept of specific mutual information, to guide the user to explore the interesting values among two variables. Lu and Shen [57] apply information entropy to select a subspace between two variables that contain the most independent information, and then explore interesting features from the selected subspace. They provided an effective system for users to interactively explore the multivariate dataset. For isosurface analysis, Bruckner and Möller [13] proposed the *isosurface similarity map* based on mutual information for automatically selecting salient isosurfaces in a volumetric dataset. Haidacher et al. [35] extended the concept of isosurface similarity map to multimodal data.

2.3 Local Distribution-Based Features

A local distribution-based feature describes the statistics in the neighborhood region of a voxel. Given a neighborhood size, the local distribution of a voxel is obtained by calculating the frequencies of scalar values in the voxels' neighborhood region. The frequencies are typically described by a histogram, where the scalar value domain is divided into a number of small value ranges (i.e., bins), and the occurrence of values falling into each bin is counted.

2.3.1 Applications of Local Distribution

The usage of features represented by the histogram has been applied in diverse fields such as computer vision, medical image visualization, and scientific data visualization. Applications in the field of computer vision include object detection [30] and tracking [47], face detection [72], and human detection [27] and tracking [5]. For data visualization,

Laidlaw et al. [52] proposed the partial volume classification to identify the material composition in a local region. Lundström et al.[58] proposed the partial range histogram (PRH) to identify the value range of the interesting material and then applied the PRH for voxel classification. The classification result is then integrated into a transfer function design to provide detailed information in regions of interest. Gu et al. utilize a block level histogram to track features in time-varying data [32]. Thompson et al. [85] proposed the use of the hixel, which stores the histogram per voxel or per block and applied the local histogram to compute fuzzy isosurfaces to identify the locations of material boundaries.

2.3.2 Local Distribution Extraction Approaches

In the past decades, several approaches have been proposed to efficiently extract local histograms at arbitrary positions in a scalar volume dataset. Porikli [70] proposed integral histogram, an extension of the summed area table, which extracts a local histogram in constant time so that it has been applied in many works such as object tracking [1]. Due to the requirement of expensive storage overhead for integral histograms, several techniques have been adopted. Lee et al. [54] proposed a wavelet-based compression technique for integral histograms. Chaudhuri et al. [17] utilized the decomposition scheme and similarity-driven indexing to reduce the storage cost. However, their storage overhead can still be larger than the original data size. To extract local histograms at all data points, Perreault and Hebert [69] proposed an approach only to scan through non-overlapping regions and update the histogram from the previous query region by maintaining column histograms. Based on their method, Sizintsev et al. [79] proposed the distributive histogram for extracting local histograms at all data points, which can achieve higher performance than the conventional algorithm in 2D cases. Applying these works to the local distribution-based feature search,

they still have to scan through all the voxels in the data, which becomes costly in large 3D volume datasets.

2.4 Multi-field Data Analysis and Visualization

A process of understanding and analyzing multi-field datasets is to discover the complex relationship between different fields (variables) and applying visualization techniques. Sauber et al. [74] proposed an approach based on local correlation field to explore the relationships among multiple fields. Nagaraj et al. [61] proposed a gradient-based comparison measure for multi-field data. Wang et al. [88] explored the causal relationships by measuring information transfer among variables in time-varying multivariate datasets. However, these works only explore the average relationships among variables but ignore the interactions among different values in different fields. Recently, several approaches have been proposed to solve this issue. Biswas et al. [6] proposed an exploration framework based on information theory to guide the user to determine the interesting values among the variables. Liu and Shen [55] introduced the concept of informativeness and uniqueness from the field of social network analysis to explore the scalar relationships between different variables, and developed an interactive interface to assist users to identify the interesting scalar values in the multi-field datasets. Lu and Shen [57] proposed an interactive visualization system for exploring interesting features in the multivariate dataset. Furthermore, feature-based analysis and visualization is applied to multi-field datasets as well. Jänicke et al. [41] applied local statistical complexity to identify unique features for analyzing time-varying multi-field dataset. Johnson and Huang [42] developed boolean predicate clauses to define features based on individual local distribution for each field. Wang et al. [90] recently collected 3D SIFT features from each field to construct feature descriptions. These

works majorly focus on features from individual fields separately and ignore the features encoded in a joint set of fields, which can describe more precisely the occurrences of the joint values in multi-fields datasets.

Another prevalent approach of multi-field visualization is to first reduce the number of dimensions of the original dataset since multivariate dataset usually comes with a high-dimensional attribute space. The concept of dimension reduction is to project the data points from a higher-dimensional space to a lower-dimensional space while preserving the characteristics of the original high-dimensional data. Principle Component Analysis (PCA) is one of the prominent dimension reduction approach applied to analyze and visualize multivariate dataset [63, 71]. It uses an orthogonal transformation to transform the data from the high-dimensional coordinate into a set of coordinates in a low-dimensional data space while maximizing the variability in the data. Multidimensional Scaling (MDS) is another well-known approach for multi-filed data visualization [8, 11]. It projects the high-dimensional data points into a set of low-dimensional points while maintaining the distance in the high-dimensional data space between each point. Parallel Coordinates plot (PCP) is also a notable approach to display the relationship between variables. Recently, several works combined the techniques mentioned above to analyze and visualize multi-field datasets. For example, Guo et al. [34] integrated PCP and MDS plot for exploring interesting feature by interactively determining multidimensional transfer functions.

2.5 Bitmap Index

Bitmap index is an efficient indexing data structure which takes advantage of bit-wise operations supported by the computer hardware. It can quickly respond to the value range query and has frequently been applied in query-based visualization applications [23, 73,

Table 2.1: An example of bitmap index.

Dataset		Bitmap Index			
Voxel ID	Data Value	b_0	b_1	b_2	b_3
		[0.0,1.0)	[1.0,2.0)	[2.0,3.0)	[3.0,4.0]
0	0.2	1	0	0	0
1	1.1	0	1	0	0
2	2.4	0	0	1	0
3	2.7	0	0	1	0
4	3.8	0	0	0	1
5	2.2	0	0	1	0
6	1.6	0	1	0	0
7	0.7	1	0	0	0

80]. An example of bitmap index is shown in Table 2.1. In this simple example, we divide the whole data value range into smaller groups and assign each group to a corresponding bin, which is a similar concept to the histogram. In table 2.1, the dataset has 8 elements and the data value range is from 0.0 to 4.0. Here the bin width is set to 1 so that four distinct value ranges are created. Each column of this bitmap represents one value range and is stored in a *bit vector* whose length is the same as the total number of data elements. In each bit vector, a bit is set to 1 if the value of the corresponding voxel falls into the value range; otherwise, it is set to 0.

2.5.1 Compressed Bitmap Index

As the size of scientific datasets continues to grow, compressed bitmap index is frequently applied to reduce the data size overhead. Multiple compressed bitmap indexes based on the run-length encoding scheme have been proposed, such as Byte-aligned Bitmap Code (BBC) [3] and Word-Aligned Hybrid (WAH)[98, 99]. WAH32 compression is used

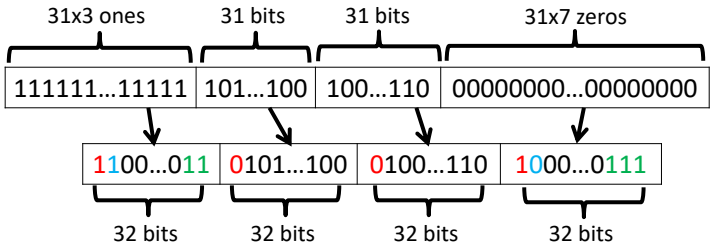


Figure 2.1: An example of WAH32 encoding scheme. Upper row: an uncompressed bit vector with 31×12 bits. Lower row: a WAH32 compressed bit vector stored in 32×4 bits.

in our entire work since it achieves better performance of logical operations than most of the other compressed bitmap indexing algorithms [97].

WAH32 scheme partitions a bit vector into several groups and each group contains 31 bits. The bit groups are categorized into two types: fill group and literal group. The fill group contains identical bit in 31 bits and is categorized into two types: 0-fill (all bits are 0) and 1-fill (all bits are 1); The literal group contains both 0 and 1 in 31 bits. In WAH32, both fill group and literal group are stored using a word (32 bits). The word storing fill group is defined as a fill word and the word storing literal group is defined as a literal word. The first bit in a word is used to indicate which type of word, 1 for fill word and 0 for literal word. The remaining 31 bits in the word store different information for the two types of word. For the literal word, the remaining 31 bits are the original literal bit. For the fill word, the second bit is used for differentiating the type of fill group (0 for 0-fill and 1 for 1-fill). The remaining 30 bits, which are called fill-length bits, encode the count of consecutive groups of identical fill bit (run length).

Figure 2.1 shows an example of WAH32 compression. In this example, the uncompressed bit vector (upper row) has three 1-fill groups, seven 0-fill groups, and 2 literal groups. The lower row is the compressed bit vector using the WAH32 encoding. The red

bit is the indication of word type, 1 for fill word and 0 for literal word. The blue bit only in fill word is used to indicate 1-fill or 0-fill. The green bit in fill word is the run length encoding in binary format. By applying the WAH32, the first 31×3 bits and the last 31×7 bits can be reduced to two words.

Since a series of 0s or 1s is frequently encountered in the bit vector, WAH32 only encodes the counts of consecutive 0s or 1s. Since the length of the vector describing this encoding count is much smaller than the length of the bit vector, the size of the bitmap and the time to access a bit can be reduced. In terms of query response time, our research benefits from the compressed bitmap in two ways. First, it reduces the I/O time by loading shorter vectors. Second, the time to find the bit that is assigned to 1 in a bit vector becomes faster when decoding a shorter vector.

2.6 Statistical Sampling

Applying statistical sampling is a common approach to create a data summarization for reducing the data storage overhead. Simple random sampling (SRS) is a basic and frequently used approach that randomly selects a certain percent of data points out of the raw data. However, it is not designed for preserving the characteristics of raw data, so the subsamples is generally not a sufficiently accurate representation of the raw data, especially when the sample size is small. The stratified random sampling (StRS) [25] is a commonly used approach to improve the simple random sampling (SRS) by preserving the characteristics of the raw data. The concept of StRS is to subdivide the raw data into several strata and randomly draw the same percentage of samples form each stratum. A common way of creating strata is to partition the data space into multiple non-overlapped spatial regions. Compared to SRS, the representativeness of the subsampled data drawn by StRS can be

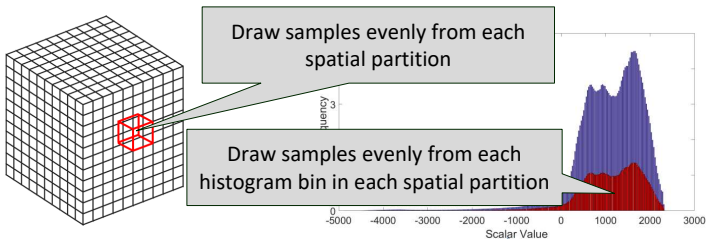


Figure 2.2: The concept of AStRS.

significantly improved by preserving the spatial distribution of the raw data. This stratified sampling is also naturally suitable for large-scale scientific simulation since the simulated data generated region by region (each region is computed by a processing unit) can be thought of separating the data into spatial strata. Woodring et al. [95] applied KDTree-based stratified sampling to store level-of-detail sample data in a particle simulation for interactive analysis and visualization. However, data reorganization is necessary before performing sampling which increases the computation time of creating representative data.

An advanced stratified random sampling (AStRS) is to divide the samples into several strata based on multiple data properties, which intends to preserve multiple characteristics of raw data simultaneously. One of the AStRS proposed by Su *et. al* [81] is to sample data based on the spatial location and data value in order to preserve both spatial distribution and value distribution simultaneously. The concept of this AStRS is shown in Figure 2.2. They leveraged bitmap indexing to subdivide data samples into several value ranges (bins) first. A bit vector is created as described in Section 2.5. Each bit vector is then subdivided into several non-overlapped sectors with equal length which is the same concept as spatial partitions. To create subsampled data, even samples are drawn from each sector in a bit vector and the same sampling percentage is applied to all other bit vectors.

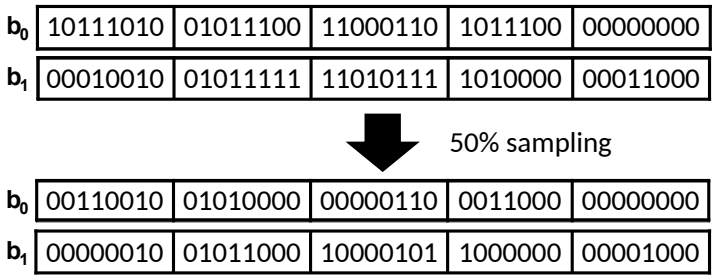


Figure 2.3: An example of performing the stratified sampling on two bit vectors representing two bins b_0 and b_1. Upper row: two bit vectors with full samples. Lower row: two bit vectors after drawing 50% of the samples from each block.

Figure 2.3 shows an example of the stratified random sampling using bitmap indexing. Suppose 50% data are drawn from two bit vectors representing two bins b_0 and b_1, which are shown in the upper part. For each bit vector, we partition the space into 5 blocks and draw samples from each block. To perform drawing samples on a bit vector, we randomly turn off 50% of the bits in each block. The sampled bit vector is shown in the lower part. Although the stratified sampling preserves multiple characteristics of raw data simultaneously, it ignores the data complexity and mainly draws the same amount of samples from each stratum. This could lead to loss of information in the regions with high data complexity but redundant information in the regions with low data complexities.

Chapter 3: Discovering and Evaluating Representative Isosurfaces based on Information-Theoretic Metrics

3.1 Introduction

Displaying isosurfaces is a common method for visualizing scalar fields. Given a scalar value, the isosurface reveals the spatial distribution of the corresponding value. In order to efficiently explore a large-scale dataset, scientists usually only focus on some values they are interested in and visualize the corresponding isosurfaces. Selecting a set of isosurfaces to summarize information in the data is also a common approach to reduce the data exploration time. However, to determine salient isosurfaces automatically from the dataset is crucial. Several approaches have been designed to guide the selection of salient isosurfaces in the past decades. These approaches exploit the statistical features of the dataset [4, 14, 28, 75], geometric features [46], topological features [15, 16, 101], as well as features that can be derived from distance transforms [13]. While these methods can select salient isosurfaces based on certain specific criteria, not much attention has been put on developing a quantitative measure to determine the representativeness of the selected isosurfaces for the entire scalar field. More specifically, it is difficult to know if any spatial subregions or value intervals are under-represented.

In this Chapter, we present an information-theoretic approach for selecting and evaluating isosurfaces from a scalar field. Our approach is based on the following idea: given two isosurfaces that enclose a subvolume, if the isosurfaces within this region change smoothly from one bounding isosurface to the other, then these two isosurfaces are considered representative for this subvolume, and hence no additional isosurfaces within the subvolume are needed for further analysis. To implement this idea, given a pair of isosurfaces, we first determine if the enclosed region satisfies such a smooth condition. We apply a level-set method [12] to morph from one isosurface to the other and generate the intermediate surfaces. From these morphed surfaces, we determine if they are representative for any true isosurface in the enclosed region by sampling the scalar field and checking the value distribution of the samples. If all the samples on the morphed surface have the same value, then this surface is indeed one of the true isosurfaces. On the other hand, if the sample values exhibit larger variance, i.e., the entropy value for the samples is higher, then this surface is not aligned well with any true isosurface in the field. When a majority of the morphed surfaces in the enclosed region are not aligned well with the true isosurfaces, we consider that the subvolume does not satisfy the smooth condition. In this research, we choose the level-set method to perform morphing between two isosurfaces. This is because it is a voxel-based approach, so no explicit surface parameterizations and vertex matching are required when performing surface morphing. Also, the level-set method can be applied to the two end surfaces that do not have the same topology [64].

To evaluate the smooth condition for an enclosed region between two pre-selected isosurfaces, we employ information theory to perform a quantitative measurement. We create a 2D map called *isosurface information map* by stacking up the distributions computed

from the samples on the morphed surfaces, which are estimated as a sequence of 1D histograms. From the isosurface information map, we compute the *normalized conditional entropy* to indicate whether the isosurfaces are sufficient to represent the data in the enclosed subvolume. We can also compute the *specific-conditional entropy* to measure how well each isosurface is being represented by the morphed surfaces. From the measurements, we are able to determine whether the pre-selected isosurfaces are representative for the subvloume and whether/where more isosurfaces should be added.

Our research has several contributions. First, we present a novel quantitative approach to evaluate the representativeness of the pre-selected isosurfaces using surface morphing and information theory. We develop a technique called isosurface information map, allowing us to measure the information content in a subregion enclosed by two isosurfaces. Based on the quantitative evaluation approach, we design an algorithm for automatic selection of isosurfaces to facilitate more effective scalar field analysis.

3.2 Background

3.2.1 Surface Morphing

Surface morphing has been studied in decades by computer graphics researchers. For 3D surfaces, the algorithms can be categorized into surface-based methods and volume-based methods [12]. The surface-based morphing methods [44, 66] model the input surfaces as 3D polyhedra and create corresponding vertices between the initial and target surfaces, and then interpolate the vertices between them to morph the surface. The volume-based morphing methods, such as the level set methods [12, 64], represent the domain by a 3D function where the surface to morph becomes an isosurface of a known value (usually 0). Then the surface is updated by adjusting this 3D function. The main superiority

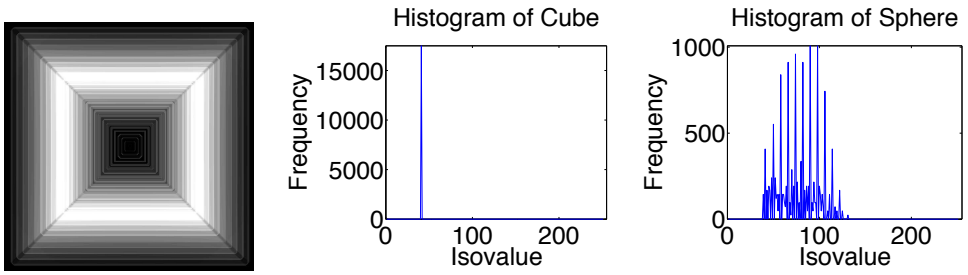
of the surface-based methods is that it only has to visit the vertices on the mesh, while the volume-based methods need to visit the entire 3D space. However, the surface-based methods need to map vertices between two meshes, which can be non-trivial process for surfaces with different topologies. In contrast, the volume-based methods do not require the mapping between the surface vertices. In this research, we apply the level-set method to achieve surface morphing since it can easily handle input surfaces with different topology, which can happen to arbitrary pairs of isosurfaces.

3.3 Method

Our isosurface evaluation algorithm is based on the idea of using the level-set method to generate intermediate surfaces from a pair of isosurfaces, and determining if the intermediate surfaces match well with the true isosurfaces in the spatial subregion. In Section 3.3.1, we describe the proposed distribution-based surface comparison method. Then in Section 3.3.2 we briefly describe the level-set method used in our work.

3.3.1 Distribution-based Surface Comparison

Given a surface, if it is an isosurface of value h in the scalar field, only the value h will be encountered when sampling the scalar field on the surface points. In other words, if we look at the distribution of the sampled values, only value h will have non-zero probability. On the other hand, if this surface intersects with multiple isosurfaces, the sampled values from the surface will have a wider value distribution. Based on this idea, we can use the value distribution of the samples on the surface to determine whether this surface is aligned well with any isosurface in the scalar field. Shannon's entropy [78] computed from the samples' distribution can be used to evaluate the alignment quantitatively. As the distribution is usually sampled in a discrete form such as histograms, we treat the scalar



(a) (b) (c)

Figure 3.1: Distribution-based shape comparison for isosurfaces. (a): A test scalar field where the isosurfaces form a layered structure of cubes. (b): The distribution of scalar values on a cube. (c): The distribution of scalar values on a sphere.

variable as a discrete random variable where the value interval of each histogram bin is an outcome of the variable. Given a discrete random variable X with probability distribution function $p(x_i)$ for outcomes $x_i, i = 1 \ldots N_x$, its entropy is defined in Equation 3.1:

$$H(X) = -\sum_{i=1}^{N_x} p(x_i) \log_2 p(x_i) \tag{3.1}$$

An important property of entropy is that its value is maximized if $p(x_i)$ are the same for all outcomes or minimized if only one outcome has non-zero probability. As a result, a low entropy for the samples of a surface means the surface is well aligned with an isosurface, and a large entropy means poor alignment. Figure 3.1 shows an example where the isosurfaces of a scalar field in Figure 3.1 (a) form layered cubes of distinct sizes. If we sample this scalar field on the face of a cube , the distribution will have a very narrow range of values with non-zero probability, shown in Figure 3.1 (b). On the contrary, suppose a sphere exists in this scalar field, the distribution of the samples on the surface of the sphere will have a wider value range with non-zero probability, shown in Figure 3.1 (c).

Following this idea, we can evaluate multiple surfaces based on the sampled data values in a scalar field. Because closed isosurfaces usually form an onion-like layered structure, given a surface that is aligned with an isosurface, we can deform this surface along its normal direction to see whether the deformed surface still aligns well with other isosurfaces. During the deformation, if the value distribution on the morphed surface always has a narrow support of non-zero probability or low entropy, we can say that an interval of isosurfaces shares a similar shape.

We can extend the idea of surface morphing and scalar evaluation to two boundary isosurfaces. Given two non-intersected surfaces which enclose a subvolume, we can morph one surface to the other via a sequence of local deformation. If the computed value distribution for each intermediate surface always has a narrow interval of non-zero probability, the shape of isosurfaces in the enclosed subvolume between the two boundary isosurfaces can be easily inferred without checking the actual isosurfaces. In such a case, the two isosurfaces are sufficient to represent the data in the enclosed subvolume.

3.3.2 Level-set-based Surface Morphing

Among the volume-based methods, we choose the level-set method presented by Breen and Whitaker [12]. The first level-set method was initially proposed by Osher and Sethian [64], which was designed to morph a single surface, while Breen and Whitaker's method can morph one surface S_{init} to the other S_{target}. The fundamental idea of the level-set method is to keep updating the scalar field such that the isosurface of value 0 is the resulting morphed surface. In the level-set method, the scalar field $\phi(\mathbf{x},t)$ is updated by solving the following partial differential equation [65]:

$$\frac{\phi(\mathbf{x},t+\Delta t)-\phi(\mathbf{x},t)}{\Delta t}+|\nabla\phi||\vec{F}|=0 \tag{3.2}$$

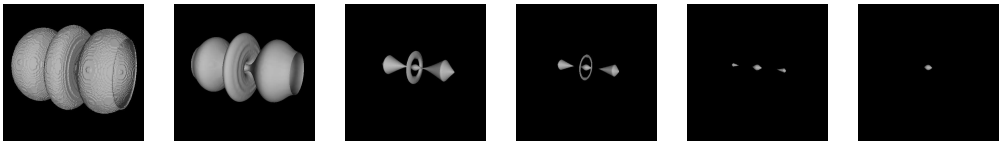
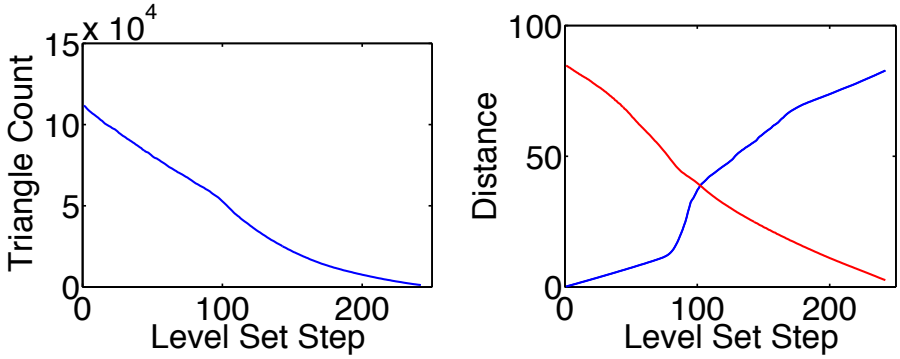


Figure 3.2: Level-set evolution example. Six morphed surfaces are shown when morphing from the isosurface 1 to isosurface 100 of the dataset *HydrogenAtom*.

Here $\mathbf{x}$ represents the spatial position and t represents the time during the surface deformation. Δt represents the *level-set step size*. When $t = 0$, $\phi(\mathbf{x}, 0)$ is set to the distance from $\mathbf{x}$ to the surface S_{init}. In other word, S_{init} represents the 0-value isosurface in $\phi(\mathbf{x}, 0)$, and the 0-value isosurface becomes increasingly close to S_{target} as t increases. The term $|\vec{F}|$ is the *force* that guides the update of ϕ in Equation 3.2. To morph the surface toward S_{target} as t increases, Breen and Whitaker use the distance to S_{target} as the force at each voxel. This force definition allows the points farther away from S_{target} to move faster than other points closer to S_{target}, which essentially allows the morphing process to have non-uniform scaling and translation between S_{init} and S_{target}. Moreover, Δt also plays an important role during the surface morphing, so we will discuss more how to select Δt for robust isosurface evaluation in Section 3.6.1. An example of surface morphing using a simulation dataset *HydrogenAtom* is shown in Figure 3.2. Here the initial and target isovalues are of value 1 and 100, respectively. Figure 3.2 shows that the morphed surface is approaching the target surface.

To study the evolution of the level-set surfaces, we show the trend of triangle count and the distance to the initial and target isosurfaces during the surface morphing. We can see that, from Figure 3.3(a), the triangle count linearly decreases from 1.5×10^5 (initial isosurface) to 10^2 (target isosurface) as the morphing time step t increases. This is an



(a) (b)

Figure 3.3: The evolution of the level-set surface using the test dataset *Plume*.(a): Triangle count of the surface at each morphing time step. (b): The distances from the morphed surfaces to the initial isosurface (blue) and the target isosurface (red)

indication that the morphed surface is stably changed between the two input isosurfaces. Besides the triangle count, we compute the root mean square (RMS) distance from the morphed surface to the two input isosurfaces. The RMS distances to both the initial surface and the target surface are shown in Figure 3.3 (b). As t increases, the distance to S_{init} becomes larger, indicating that the morphed surface is moving away from the initial surface. In the meantime, the distance to the target surface decreases, implying that the morphed surface is getting closer and thus more similar to the target isosurface.

3.4 Isosurface Information Maps and Information Measurement

Based on the distribution-based shape comparison and level-set-based surface morphing, this section describes how we use information theory to compare the level-set surfaces and the true isosurfaces. Given a set of pre-selected isosurfaces to evaluate, our approach first constructs an isosurface information map. Hereafter the input scalar field is

denoted as $X(\mathbf{x})$, and the set of isovalues to evaluate is denoted as $Y = y_1, \ldots, y_{N_y}$ where $y_1 < y_2 < \cdots < y_{N_y}$, which essentially divides the value range into $N_y - 1$ intervals.

3.4.1 Isosurface Information Maps

We first describe the isosurface information map for a subvolume between two isosurfaces. Given two isovalues y_j and y_{j+1}, we morph the bounding isosurface of isovalue y_j to that of y_{j+1}. During the morphing process, the value distribution on the deformed surface is estimated by sampling the scalar field into a histogram $h_s[x_i]$ where s is the level-set step number, and x_i represents the value interval for the i-th bin. To compute $h_s[x_i]$, our approach samples the level-set surface and counts the samples whose values fall into the interval x_i.

As the density of the samples determines the accuracy of the histogram, the sample locations should be carefully selected in order to achieve a balance between quality and efficiency. Since the marching cubes algorithm is applied to compute the zero isosurface from $\phi(\mathbf{x})$ during the surface morphing, our sampling process can take advantage of the small triangles generated in each cell by the marching cubes algorithm. For those triangles, the distance between two connected vertices on the morphed mesh is shorter than the length of the diagonal line of the cell. Since the distance is bounded and the scalar field is assumed to vary linearly within each cell, our current implementation for regular Cartesian grid datasets takes the vertices of the surface as the sample locations for building the histogram $h_s[x_i]$. It should be noted that our method can easily adopt approaches such as the contour spectrum [4] or a more general approach recently presented by Duffy et al. [28] to estimate the distribution.

By collecting $h_s[x_i]$ from all the morphed surfaces, we can create the 2D isosurface information map for the interval $[y_j, y_{j+1}]$ from the histogram $h_j[x_i, s]$. The isosurface information map is computed by normalizing the joint histogram. The information map is denoted as $p(x_i, s)$, where

$$p(x_i, s) = h_j[x_i, s] / \sum_i \sum_s h_j[x_i, s] \tag{3.3}$$

It can be seen that the isosurface information map $p(x, s)$ is the joint distribution between the isosurface of value x and the level-set surface s. Based on the isosurface information map, we utilize two information measurements to evaluate the similarity between the level-set surfaces and the true isosurfaces in the subvolume.

3.4.2 Identifying Under-represented Isosurfaces using Specific Conditional Entropy

In Section 3.3.1, we explain the approach of determining whether a level-set surface is aligned well with a true isosurface based on the value distribution of the samples on the surface. In this section, we extend the concept to identify the least represented isosurface in a given scalar interval bounded by two isosurfaces using the isosurface information map proposed in Section 3.4.1. This approach forms the foundation for our isosurface selection algorithm described in Section 3.5.

Our identification of the least represented isosurface is based on the concept of *specific conditional entropy*. Given two discrete random variables X and Y with outcome $x_1, \ldots, x_{N_x}$ and $y_1, \ldots, y_{N_y}$, and a joint distribution $p(x, y)$, we can compute the marginal distribution $p(x_i) = \sum_{j=1}^{N_y} p(x_i, y_j)$ and $p(y_j) = \sum_{i=1}^{N_x} p(x_i, y_j)$. The specific conditional entropy $H(Y|X = x_i)$ is the conditional entropy when X has a specific value $X = x_i$, and is computed using Equation 3.4:

$$H(Y|X=x_i) = -\sum_{j=1}^{N_y} p(y_j|x_i)\log_2 p(y_j|x_i) \tag{3.4}$$

where $p(y_j|x_i) = p(x_i,y_j)/p(x_i)$ is the conditional probability of $Y = y_j$ given $X = x_i$. Like entropy, the specific conditional entropy value is maximal if $p(y_j|x_i)$ has the same probability for all y_j.

Wile computing the specific conditional entropy, we take the surface generated at the level-set step s as the outcome of a random variable S that has values $1 \ldots N_s$. N_s is the number of level-set steps, also the number of intermediate surfaces computed. By replacing the random variable Y above with S and the outcomes $y_1, \ldots, y_{N_y}$ by $1, \ldots, N_s$, we can compute the specific conditional entropy $H(S|X = x_i)$ for each x_i utilizing the isosurface information map. If a particular x_i has a large $H(S|X = x_i)$ value, it means that the corresponding $p(x_i, s)$, $s = 1 \ldots N_s$, in the isosurface information map has a wider probability spread across multiple level-set surfaces, an indication of this isosurface x_i intersecting with multiple level-set surfaces so cannot be represented well by any of them. In other words, $H(S|X = x_i)$ indicates the quantification of the uncertainty if a true isovalue x_i is being represented by the level-set surfaces. When this value is high, the level-set surfaces cannot well describe the isosurface x_i.

It can be shown that the range of the specific conditional entropy is in $[0, log_2(N_s)]$. The specific conditional entropy $H(S|X = x_i)$ can be normalized to the range of [0, 1] as in Equation 3.5:

$$H'(S|X=x_i) = H'(S|X=x_i)/log2(N_s) \tag{3.5}$$

3.4.3 Isovalue Interval Evaluation

Besides identifying the under-represented isosurfaces, here we show how to utilize the isosurface information map to evaluate whether the two boundary isosurfaces well represent all isosurfaces in the enclosed subvolume.

Given two random variables X and Y, their mutual information [78] can be computed as in Equation 3.6:

$$I(X,Y) = \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} p(x_i, y_j) \log_2 \frac{p(x_i, y_j)}{p(x_i)p(y_j)} \tag{3.6}$$

Consequently, we can compute the conditional entropy $H(X|Y) = H(X) - I(X,Y)$ of X given Y. Conditional entropy $H(X|Y)$ means the uncertainty about X given the knowledge of Y. When $H(X|Y)$ is 0, it means that the outcomes of X can be perfectly predicted given the outcomes of Y. Conversely, if the outcomes of Y cannot help us to gain any information about the outcomes of X, $H(X|Y)$ becomes $H(X)$.

As the value range of conditional entropy $H(X|Y)$ is $[0, H(X)]$, we can normalize $H(X|Y)$ by the maximum possible value of $H(X)$, which is $\log_2 N_x$:

$$H'(X|Y) = \frac{H(X|Y)}{\log_2 N_x} = \frac{H(X) - I(X,Y)}{\log_2 N_x} \tag{3.7}$$

In our case, by treating the true isosurfaces as one random variable X, the level-set surfaces as another random variable S, we can compute normalized conditional entropy $H'(X|S)$ utilizing the isosurface information map. During the surface morphing, we know that the level-set surface s aligns well with the isosurface x_i if the value distribution of the samples on the surface s only has non-zero probability at value x_i. If each s is mapped to one isosurface, indicating that the level-set surface always aligns well with one true isosurface, the normalized conditional entropy $H'(X|S)$ will be 0. In other words, by checking whether

the normalized conditional entropy is close to 0, we can evaluate the represntativeness of the given pair of isosurfaces for the enclosed subvolume.

3.5 Information-theoretic Isosurface Selection

To provide an effective visualization of scalar fields, based on the isosurface information map introduced in Section 3.4.1 and the evaluation methods introduced in Section 3.4.2 and 3.4.3, we design an automatic isosurfaces selection algorithm.

3.5.1 Algorithm

Our algorithm takes an initial set of isosurfaces as input, denoted as S_{known}. S_{known} can be generated by user selections or by any existing approaches, for instance, the automatic algorithm presented by Bruckner and Möller [13]. We first sort the isovalues used in S_{known}, and then for each pair of consecutive isovalues y_j and y_{j+1}, we build an isosurface information map as described in Section 3.4.1. We compute the normalized conditional entropy to evaluate whether the isosurface of y_j and y_{j+1} are sufficient to represent the enclosed subvolume. If not, a new isosurface between y_j and y_{j+1} will be added. As our goal is to select an isosurface that is the most under-represented between y_j and y_{j+1}, we compute the specific conditional entropy for each isovalue between $[y_j, y_{j+1}]$ from the information map, and choose the one that has the largest specific conditional entropy.

After a new isovalue y_{new} is added, the interval $[y_j, y_{j+1}]$ is divided into two new sub-intervals $[y_j, y_{new}]$ and $[y_{new}, y_{j+1}]$. We recursively evaluate the sub-intervals and add new isosurfaces if needed. To stop the recursive insertion of isosurfaces, our algorithm has two termination conditions. The first one is to check whether the normalized conditional entropy is smaller than a given threshold. The other one is to check the similarity between the two boundary isosurfaces. Given an interval $[y_j, y_{j+1}]$, to determine the similarity between

the two boundary isosurfaces, we compute the distance from each vertex on one isosurface to the other isosurface, and then compute the RMS value of these distances. If the RMS value is smaller than a pre-defined threshold, which means the isosurface of value y_j is close enough to the isosurface of value y_{j+1}, we then stop the insertion of isosurfaces between these two isovalues.

3.5.2 Case Study: *HydrogenAtom*

Figure 3.4 shows the experiment results for the scalar field *HydrogenAtom*, which is a spatial distribution of electrons within a hydrogen atom. In this experiment, we chose the isosurface of value 1 as S_{init} and value 100 as S_{target}, which are shown in the first row of Figure 3.4. The isosurface of value 1 is the largest isosurface in the whole data domain. When sweeping the isovalues from 1 to 100, the isosurface is gradually shrinking to a small point.

Figures 3.4 (3-a) to (3-c) list the first three isosurfaces selected by our algorithm. The heat maps (Figures 3.4 (1-a) to (1-c)) are the isosurface information maps for each pass of subvolume evaluation. The x- and y- axes represent the scalar value and the level-set step, respectively. The color from blue to red represents the intensity from low to high. Figures 3.4 (2-a) to (2-c) show the specific conditional entropy from the corresponding isosurface information map, where the x- and y- axes represent the scalar value and the specific conditional entropy, respectively.

The isosurface information map in Figure 3.4 (1-a) shows that for each isovalue, its corresponding column has a wide spread of non-zero probability. This is expected because S_{init} and S_{target} are quite different, and thus the intermediate surfaces generated by the level-set method are supposed to be quite dissimilar to the true isosurfaces. The isosurface of

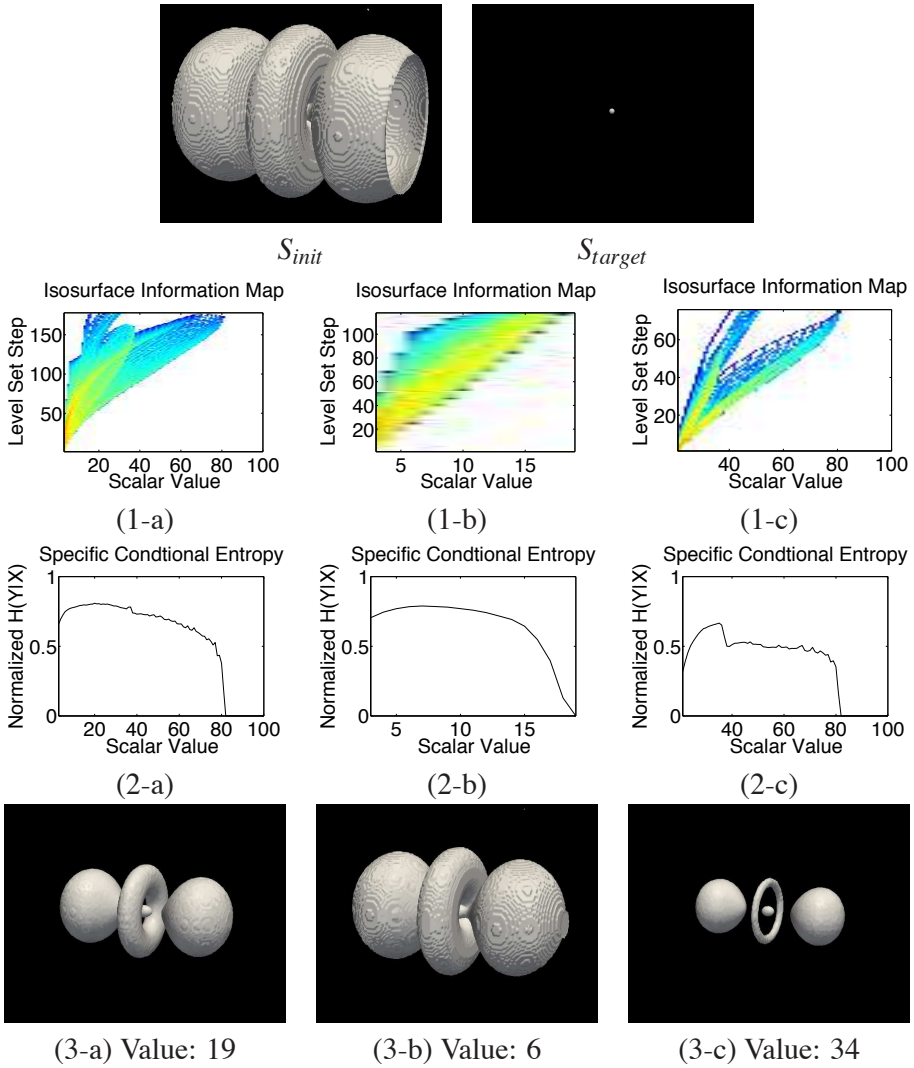


Figure 3.4: Result of dataset *HydrogenAtom*. The top row shows the initial isosurfaces S_{init} (isovalue 1) and S_{target} (isovalue 100). The columns below from left to right represent the result for the first three selected isosurfaces. (1-a)-(1-c): Isosurface information maps. (2-a) - (2-c): The corresponding normalized specific conditional entropy. (3-a)-(3-c): The selected isosurfaces.

scalar value 19 is chosen to insert because it has the largest specific conditional entropy, as shown in Figure 3.4 (2-a). After adding the isosurface of value 19, we repeat the selection

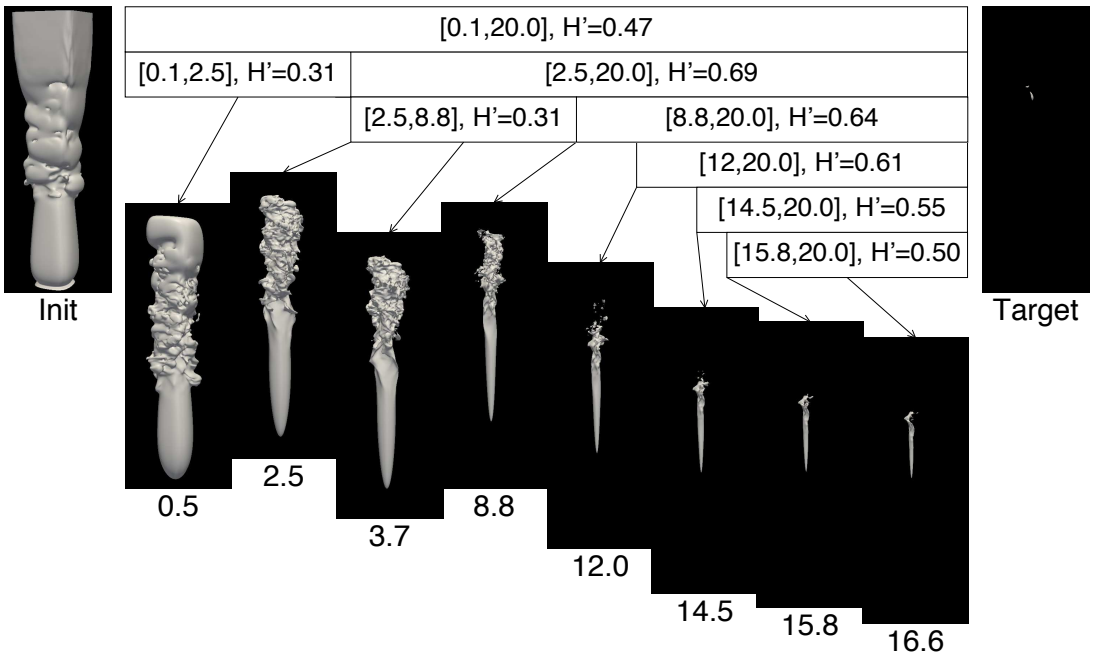


Figure 3.5: The selected isosurfaces for *Plume* from the interval [0.1, 20.0]. Each white bar represent a value interval and the normalized conditional entropy H'. Below each bar, in addition to the divided subinterval, the selected isosurface is also shown.

for the intervals [1,19] and [19,100]. The selection results for the two intervals are shown in the second and third columns, respectively, of Figure 3.4. The isovalue selected from [1,19] is 6. When sweeping across this value, the isosurface is changed from open to closed. The isovalue selected from [19, 100] is 34. This is because that the topology changes when sweeping through the isovalue 34, and thus the isosurfaces around this value will get a higher chance to be selected.

3.5.3 Case Study: *Plume*

Our second case study is done using the dataset *Plume*, a simulation of thermal plumes on the solar surface. Here we used the vector magnitude as the input scalar field, and we

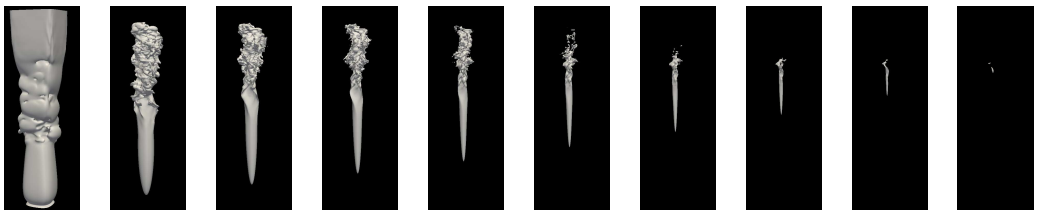


Figure 3.6: Regular sampling of 8 isosurfaces from the value interval [0.1, 20.0] of *Plume*.

have down-sampled the data by 2 into the resolution of $63 \times 63 \times 256$ cells. Our test chose the isosurface of vector magnitude 0.1 as S_{init} and that of vector magnitude 20.0 as S_{target}.

Figure 3.5 shows the experiment result. Each white bar indicates a value interval and the corresponding normalized conditional entropy (H'). Beneath each interval, the subintervals are also shown, while the subintervals with small H' or small RMS distance are hidden. The rendered images are organized such that the isosurfaces selected earlier are placed higher.

It can be seen that when the interval is divided, the normalized conditional entropy (H') of the subintervals is smaller than the original interval except for the first interval. In the end, 8 isosurfaces were selected. Among the selected isosurfaces, the one for magnitude between 2.5 and 8.8 are more turbulent in the top half of the volume. The isosurfaces of larger magnitude (for 12 to 16.6) are mainly distributed in the central region, and the size shrinks when the vector magnitude increases.

We compare our result with the output by regular sampling. Since our algorithm select 8 isosurfaces from the interval $[0.1, 20]$, the same number of isosurfaces regularly sampled in this interval are shown in Figure 3.6 for comparison. Since the first sampled isosurface is for vector magnitude 2.3, any change of isosurfaces within the interval [0.1, 2.3] is missed. Nevertheless, according to the isosurface of magnitude 0.5, which is shown in Figure 3.5,

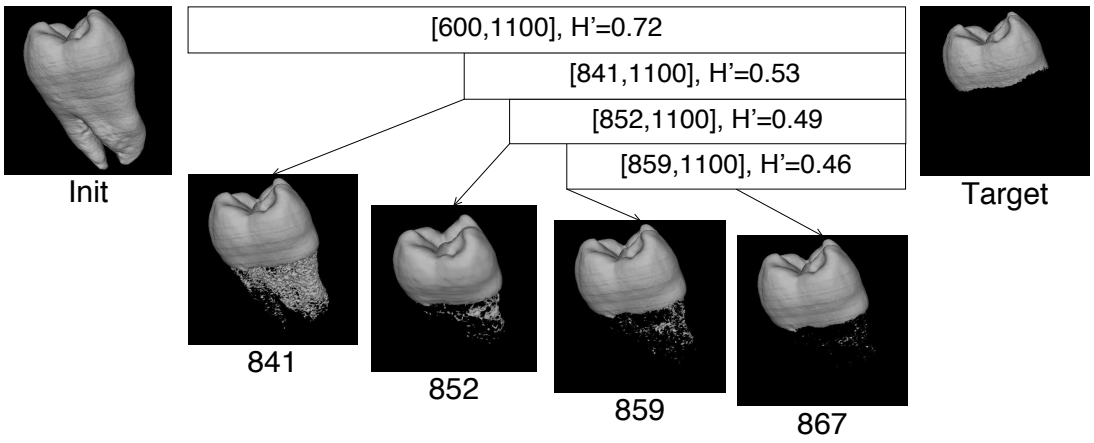


Figure 3.7: The selected isosurfaces for *Tooth* from the interval [600, 1100].

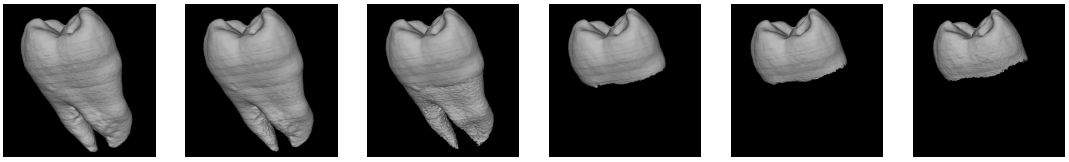
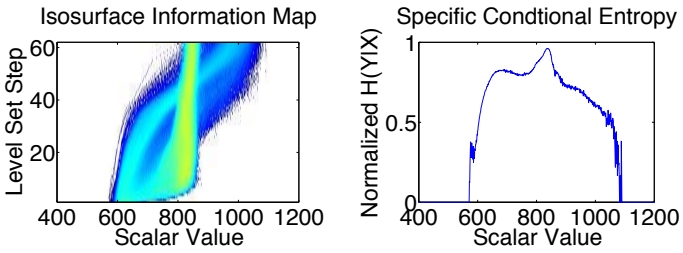


Figure 3.8: Regular sampling of 4 isosurfaces between isovalues 600 and 1100 of *Tooth*.

the turbulent flow and the outer smooth flow are mixed within this interval. While this mixing can be seen in our selected isosurfaces, it is missed by regular sampling.

3.5.4 Case Study: *Tooth*

So far the test datasets used in the previous case studies are all generated by scientific simulations. Here we show the result from a medical dataset *Tooth*. We down-sampled the dataset by half into the resolution of $128 \times 128 \times 82$. In this experiments, we chose



(a) (b)

Figure 3.9: The isosurface information map (a) and the normalized conditional entropy (b) for *Tooth*.

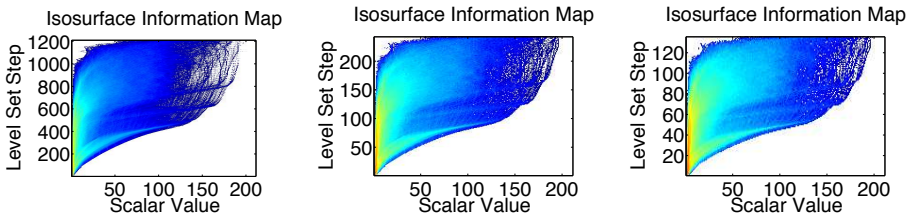
the isosurface of value 600 as S_{init} and the isosurface of value 1100 as S_{target} which are regarded as the salient isosurfaces presented in [84].

Figure 3.7 shows our selection result. Our algorithm terminated because the RMS distance is small. It can be seen that the four selected isosurfaces are concentrated in the interval $[841, 867]$. The reason is that within this value interval, the shape of the isosurfaces changes greatly. Compared to the regular sampling case shown in Figure 3.8, regular sampling cannot capture the transition of the isosurfaces without more isosurfaces being sampled. This can also be seen in the isosurface information map and the normalized conditional entropy in Figure 3.9. As a result, more isosurfaces were selected by our algorithm to illustrate the transition.

3.6 Discussion

3.6.1 Level-Set Step Size Selection

As mentioned in Section 3.3.2, the computation of level-set surfaces requires a step size Δt. A larger step size can complete the surface morphing in fewer steps, which is faster but may lead to insufficient sampling of the scalar field, while a smaller step size slows



$\alpha = 0.1$ $H' = 0.88$ | $\alpha = 0.5$ $H' = 0.84$ | $\alpha = 0.9$ $H' = 0.83$

Figure 3.10: An example isosurface information map with different CFL number α of level-set evolution for *Plume*.

down the morphing but gives a smoother morphing result. To balance the computational performance and sampling quality, our implementation controls the step size Δt at each iteration according to the *Courant-Friedrichs-Lewy condtion* [18]:

$$\Delta t < \Delta x / \max(|V|) \tag{3.8}$$

where Δx is the voxel size of the volume data and $\max(|V|)$ is the maximum velocity magnitude on the morphed surface, which is linearly proportional to the magnitude of the maximal force. Essentially, Courant-Friedrichs-Lewy condition prohibits the surface movement at any point to exceed one grid cell in one iteration, which thus defines an upper bound of the level-set step size.

To control the surface morphing speed, Δt can be further weighted by a scalar $\alpha, 0 < \alpha < 1$. Figure 3.10 shows an example of the isosurface information maps with different α. Smaller α makes a smoother morphing so that the information map is more refined than the images in Figures 3.10 (b) and (c), but more steps will be taken and thus slower. As the conditional entropy only slightly changes when α is between 0.1 and 0.9, we choose α as 0.5 in all cases in this work.

Meanwhile, when the morphed surface is close to the target surface, we should adjust the level-set step size in older to conform the morphed surface to the target surface. Our implementation follows the stop criteria suggested by Breen and Whitaker [12]. In each iteration, the root-mean-squared (RMS) distance between the morphed isosurface and target isosurface is computed. Once the RMS distance is smaller than a threshold, the level-set method terminates.

3.6.2 Performance

This section describes the computational performance of our algorithm. Our approach was implemented in C/C++ and tested on a machine with MS Windows 7 OS, an Intel Core 2 Duo E6750 CPU, 8GB system memory, and an nVidia GeForce GTX 460 GPU with 1GB of texture memory. We implement and modify Breen and Whitaker's level set method [12] for our algorithm. The modified level-set method includes three stages: collecting the distribution of scalar values from the original field on the zero isosurface, updating the scalar field ϕ, and computing the distance from the morphed surface to the target surface.

Two strategies are used to accelerate our algorithm. The first one is to use a simplified marching cubes implementation to efficiently collect the distribution on the zero isosurface. Conventionally, marching cubes algorithm computes the vertices on the cell edges and create the triangular mesh. Nevertheless, we only need the values on the mesh vertices to form the distribution, and the connectivity information is not necessary. Therefore, we can only check each grid edge to see whether it intersects the zero isosurface. If so, the value on the intersected point is computed via linear interpolation and the distribution can be updated directly without building the mesh.

Data Name	Value Interval	#	Prep	ϕ	MC	Dist
H.Atom	1-100	179	8.7	8.9	4.2	1.0
Plume	0.1-20	241	5.2	11	2.7	1.4
Tooth	600-1100	69	3.2	0.4	0.9	0.9

Table 3.1: Performance (in seconds) of surface morphing for different value intervals. The column (#) records the numbers of level set steps. The stages includes preprocessing (*Prep*), update of ϕ (ϕ), marching cubes (*MC*), and distance computation (*Dist*, which was measured on GPUs). The data resolutions from the top to the bottom are 128^3, $63^2 \times 256$, and $128^2 \times 82$. Here *H.Atom* means *HydrogenAtom*.

The second strategy is using GPUs to accelerate the distance computation. As the distance computation involves the comparison from all voxels to all isoruface vertices, it will be time-consuming on CPUs. Our GPU-based distance computation is based on nVidia CUDA. Each vertex on the target surface is assigned to one CUDA thread and the vertices on the morphed surface are divided into batches. For each batch of vertices, our implementation first sends this batch to the constant memory on the GPU device, and then invokes the CUDA kernel at each thread to compute the minimal distance from its vertex to this batch. Then each thread records the minimum distance over all batches. The benefit of this implementation is the utilization of constant memory. Because the constant memory is cached, the memory access overhead on GPUs is hugely reduced, leading to 60 times speed-up for distance computation.

The computation time of one-iteration level-set evolution is shown in Table 3.1. In the future, we will further improve the performance by parallelizing the computation of updating of ϕ, which is executed at each voxel.

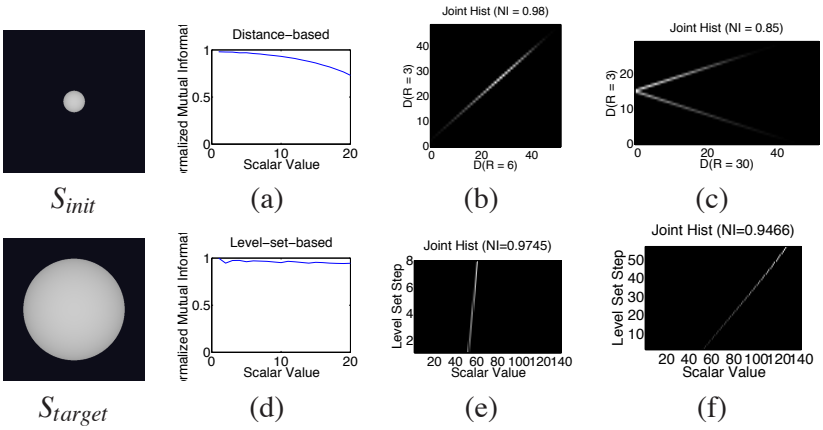


Figure 3.11: Isosurface similarites computed based on Bruckner and Möller's method [13]. (a): The isosurface similarity [13] between 20 sampled isovalues and S_{init}. (b) and (c): The joint distributions between the distance fields of S_{init} and spheres of radiuses 6 and 30, respectively. (d): The normalized mutual information between the 20 samples and S_{init}. (e) and (f): The isosurface information maps for spheres of radiuses 6 and 30, respectively.

3.6.3 Comparison with Isosurface Similarity Maps

A similar work to ours is the *isosurface similarity map* presented by Bruckner and Möller [13]. Given two surfaces says S_A and S_B, their approach computes the Euclidean distance fields D_A and D_B to the two surfaces, builds the joint distribution of $D_A(\mathbf{x})$ and $D_B(\mathbf{x})$ from all voxels $\mathbf{x}$, and then use the mutual information from the joint distribution to compute the similarity between S_A and S_B.

While both our isosurface information map and their isosurface similarity map are derived from the concept of isosurface shape comparison, their metric is more sensitive when two surfaces have different scales. The reason is that their similarity measurement is originally designed as a cost function for surface registration [39], which needs to assign a higher cost if one surface is transformed from the other via scaling/translation/rotation. This property makes their approach produce a larger distance when two surfaces have similar shapes,

but at different scales. On the other hand, in our method we consider the information content within an interval volume low when the bounding isosurfaces have similar shapes, i.e., one can be smoothly morphed from the other, even when they have quite different scales.

To verify this, we conducted an experiment by testing a 3D volume where the value of each voxel is its distance to the center of this volume. Thus the isosurfaces in this volume are concentric spheres. Then we selected two initial isosurfaces S_{init} and S_{target}, as respectively shown in Figures 3.11. Because isosurfaces in this volume are spheres of different sizes, the similarities between each pair of isosurfaces should be high.

Figure 3.11 (a) shows the isosurface similarities between S_{init} and each of the 20 sampled isosurfaces within S_{init} and S_{target} based on Bruckner and Möller's approach. Meanwhile, Figure 3.11 (d) shows the normalized mutual information between S_{init} and each sampled isosurface based on our approach. It shows that the measurement computed by our approach can better reflect the expected shape similarity.

To explain why the isosurface similarity maps are more sensitive to different scalings, Figure 3.11 (b) shows the joint distribution between the distance fields computed from a sphere of radius 3 and that of radius 6. The joint distribution has a diagonal pattern, which means a one-to-one mapping. In contrast, Figure 3.11 (c) shows the joint distribution between the distance fields from the sphere of radius 3 and that of radius 30. It can be seen that among the voxels that have a fixed distance to the sphere of radius 30, there are two possible different distances to the sphere of radius 3. In other words, even though the sphere of radius 3 and 30 have a similar shape, the similarity computed based on Bruckner and Möller's method can be low.

3.7 Summary

In this Chapter, we present an information-theoretic approach to select and evaluate the salient values/isosurfaces in a scalar field. Given two isosurfaces that enclose a subvolume, our approach uses level-set-based surface morphing to check whether the true isosurfaces in the subvolume can be generated by smoothly morphing from one isosurface to the other. During the morphing, our approach collects the distribution of the samples on the intermediate surfaces and form an isosurface information map. From the map, we apply information theory to measure how well the intermediate surfaces describe the true isosurfaces in a subvolume, and the remained uncertainty of each true isosurface. Based on these measurements, an automatic isosurface selection algorithm is also presented.

Chapter 4: Efficient Local Histogram-Based Feature Search via Bitmap Index

Representing features by local histograms has been utilized in many volume analysis and visualization applications. It has been shown that histograms in local regions can be used to identify material boundaries [85], predict material composition in a local area [52], or improve transfer function design [58]. In the applications of histogram-based feature, it is a crucial problem that search for the regions that have similar histogram due to the high computational complexity. However, this has not drawn much attention in the past works. To find the local features that match a user-defined local histogram requires to compute the local histogram at each region and compare it with the target one. This time-consuming exhaustive search prohibits the aforementioned applications from being practical, especially in the cases with large-scale dataset. Recently several algorithms have been proposed to efficiently retrieve local histogram. For example, the integral histogram proposed by Porikli [70] is able to retrieve the histogram from an arbitrary axis-aligned region in real time. However, the tremendous storage overhead to store the integral histograms still prohibits the applications for large or high dimensional dataset. With much fewer memory requirements, Sizintsev et al. [79] proposed the distributive histogram, which reduces raw data accesses by only updating the histogram in non-overlapping regions when scanning through the data domain. Although significant performance speedup is achieved compared

to the conventional exhaustive search method, their method still have to scan through the entire data space for each feature query, which is a concern when the size of the data is large.

In this Chapter, we present an efficient algorithm utilizing bitmap index to locate voxels whose local histograms in their neighborhood match the user-defined target histogram. From our observation, the user-defined value range is usually a small subset of the entire data value range. We also observed that only those voxels whose neighboring points have values within the chosen value range could be the possible candidates that satisfy the search criteria. Therefore, we propose a local histogram-based feature search algorithm that only scans through those voxels and their neighborhood regions, which can significantly reduce the search space. In this research, we leverage the bitmap index to quickly locate data points whose values fall into the target histogram bins. With efficiently identifying the voxel location, we propose an approach based on the concept of voting scheme to collect frequency counts in each of the target histogram bins. By iteratively comparing frequency counts for each bin, we gradually reduce the search space by excluding unqualified voxels whose frequency counts do not match the target histogram, and thus much higher performance can be achieved for local histogram search. Experiments show that our algorithm is faster than traditional methods and past works in most of the test cases.

The contributions of this research are:

1. We propose a novel local histogram search algorithm without exhaustively scanning through the entire data.

2. We utilize bitmap data summarization to achieve efficient query-driven analysis and visualization.

3. We develop a system that provides users to interactively define histogram-based features and interactively explore the feature search result.

4.1 Problem Statement and Algorithm Overview

4.1.1 Problem Statement

The exploration of a volumetric dataset usually requires a detailed inspection of local features, which can often be described by local histograms. A local histogram at a voxel is constructed by collecting frequency counts of values in the voxel's neighborhood region. The goal of this work is to efficiently locate the voxels whose local histograms match the user-defined histogram without exhaustively searching in the entire volume. Before the search, the user inputs a *target histogram* H_T, a neighborhood size *NBR*, and a tolerance factor δ of the difference between the target histogram and the local histograms. In this work we define two types of histogram comparison metrics to evaluate the similarity between the input histogram and the target histogram H_T. The first type of the metric is a strict comparison in which we reject an input histogram if the difference of the frequency of any bin k within the user-defined value range is larger than the pre-defined threshold δ_{b_k}, which we call *single-bin-error comparison*. We formulate this comparison in Equation 4.1, where the detected voxels are defined as:

$$V_{final} \equiv \{v_i \mid abs(H_{v_i}(b_k) - H_T(b_k)) \leq \delta_{b_k}, \\ \forall i \in \{1, \cdots n\}, \forall b_k \in \text{user-defined value range}\} \quad (4.1)$$

Here, H_{v_i} represents the local histogram at voxel v_i. n is the total number of voxels in the volume, b_k represents one of the bins in the user-defined value range, and $H(b_k)$ is the bin frequency at bin b_k.

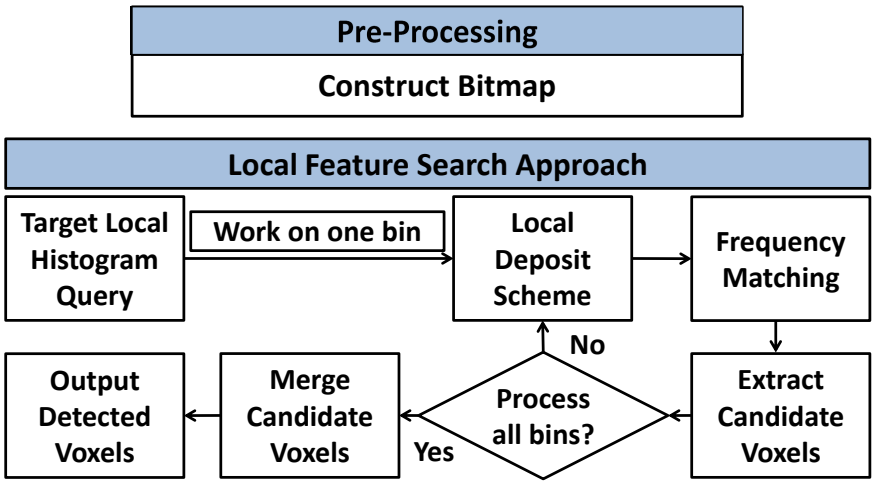


Figure 4.1: The flow chart of the proposed local histogram search approach

The second type of the histogram comparison metric is called *sum-of-error comparison*, where we reject an input histogram if the sum of the frequency errors from all the bins within the user-define value range is larger than the pre-defined error threshold δ_{sum}, as formulated in Equation 4.2:

$$V_{final} \equiv \{v_i \mid \sum_{b_k} f_{err}(H_{v_i}(b_k), H_T(b_k)) \leq \delta_{sum}, \\ \forall i \in \{1, \cdots n\}, b_k \in \text{user-defined value range}\} \quad (4.2)$$

Here $f_{err}(H_{v_i}(b_k), H_T(b_k))$ represents a bin-to-bin error between two distributions H_{v_i} and H_T. Several bin-to-bin distance metrics have been developed for evaluating the similarity between two histograms, such as $L_1 - norm, L_2 - norm, \chi^2$ statistics and Kullback-Leibler divergence (or Jeffery divergence) [59]. In this work, we apply the $L_1 - norm$ metric, and $f_{err}(\cdot)$ becomes

$$f_{err}(H_{v_i}(b_k), H_T(b_k)) \equiv abs(H_{v_i}(b_k) - H_T(b_k)) \quad (4.3)$$

4.1.2 Algorithm Overview

In this work, we propose an efficient algorithm that uses bitmap index to tackle the local feature search problem by matching local histograms. The framework of our method is shown in Fig. 4.1. Rather than scanning through the entire volume and detecting voxels that have similar local histograms to the target histogram, our method is only to search for the voxels within the value range of user-defined local features and their neighborhood. Within this search space, we perform the histogram matching algorithm based on the bin-to-bin comparison with the single-bin-error or the sum-of-error metric. A *local deposit* scheme is proposed to determine the bin frequencies at each voxel. The local deposit method is similar to a voting process which is applied to collect the corresponding value contribution from the local neighborhood of each voxel. We retrieved the voxels whose values fall into the bins within the user-defined value range from the compressed bitmap and then perform the local deposit for each retrieved voxel. After completing the local deposit, the bin frequency at bin b_k at each voxel, $H_{v_i}(b_k)$, can be determined according to its deposit count. The distance between $H_{v_i}(b_k)$ and $H_T(b_k)$ is computed for checking how similar between two histograms. Finally, we collect the final detected voxels V_{final} based on the error metric we select.

4.2 Proposed Method

In this section, we describe our proposed algorithm to efficiently search for the target histogram.

4.2.1 Local Histogram-based Feature Search Algorithm

While searching for a user-specified histogram in a dataset, the conventional approach is to go through all the voxels in the dataset; create the local histogram around the voxels and finally compare each of these local histograms with the user specified target histogram. This exhaustive search quickly becomes infeasible when the dataset is too large. In this simple search method, we make two key observations:

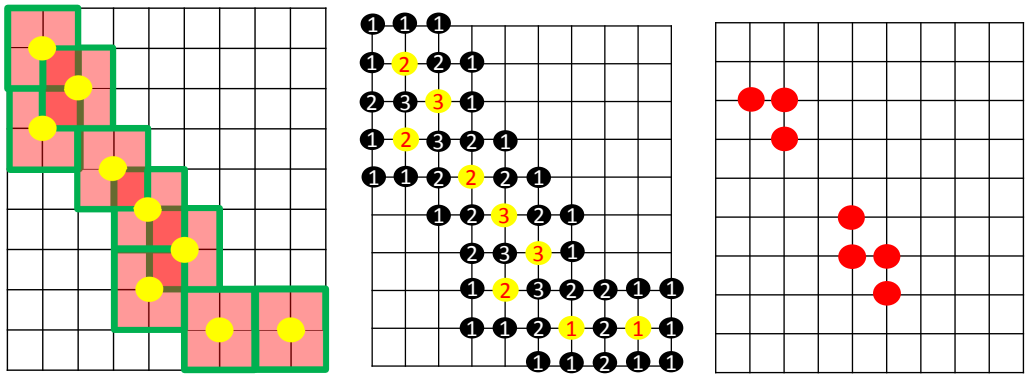
1. When the user specifies a target histogram, generally this target histogram has a smaller dynamic range compared to the value range present in the whole dataset.

2. The construction of a local histogram can be thought of inversely. Traditionally, the local histogram for each voxel is constructed by collecting data values from the voxel's neighborhood. Instead, each data element contributes one count to the local histograms of its neighboring voxels, and then the local histogram for each voxel is constructed after receiving value counts from every data elements.

Based on these two observations, we formulate a novel algorithm to identify voxels whose local histogram matches that specified by the users.

4.2.2 Histogram Matching Based on Bin-to-Bin Comparison

Once the user specifies a target histogram, we can retrieve the bins that have non-zero frequency, which are only to be compared during the histogram matching. We define these bins as *target bins*, or $\mathscr{B}$. From our observation, only the voxels whose values fall into $\mathscr{B}$ will affect the result of local histogram matching since only these voxels contribute counts to the local histograms at each voxel in the search region. These voxels are defined as *active*



(a) (b) (c)

Figure 4.2: Example of local deposit scheme for one single bin: (a): Yellow points are active voxels of the bin. Each green square region is the neighborhood of one active voxel, and the neighborhood size in this example is 3×3. Red region is the search space. (b): The number on each voxel in the search space represents a deposit count which refers to the frequency of the bin on the voxel. Yellow points represent active voxels and black points represent the rest of voxels in the search region. (c): In this example, a voxel whose frequency is equal to 3 is defined as a candidate voxel. All the candidate voxels for the bin are shown as red points.

voxels. That is,

$$V_{active} \equiv \{v_i \mid f(v_i) \in \mathscr{B}, \forall i \in \{1, \cdots n\}\} \tag{4.4}$$

where v_i is a voxel with index i, n is the total number of voxels in the dataset, and $f(v_i)$ is the corresponding value in the scalar field. Therefore, instead of exhaustively searching in the entire domain, we can reduce our search space to the active voxels and their neighborhood. Fig. 4.2 (a) shows an example of the search space. The yellow points represent active voxels of a bin of the target histogram. Each green square region is the neighborhood of one active voxel, and the red region is our search space. Compared to the whole data voxels, this reduction in the number of voxels to be processed results in a significant decrease in computation time.

Given a user specified target histogram, quickly locating the active voxels is a challenging task. With the use of the compressed bitmap indices representing the original dataset, however, this can be achieved in a much shorter time along with moderately low storage overhead. When the dataset is represented through bitmap index, the bit vectors corresponding to the bins in $\mathscr{B}$ can be extracted and used to quickly locate the physical locations of the active voxels. After locating the active voxels, the next step is to construct the local histogram to perform histogram matching, which is discussed in the next section.

4.2.3 Local Deposit

The general strategy to construct the local histogram for a voxel is to scan through all the neighborhood voxels around it and collect the counts of values to form the histogram. But the creation of the local histogram can be approached differently. Given a data element and a neighborhood size, we can locate all its neighboring voxels and put one count in the count buffer of the corresponding bin of all neighborhood voxels' histograms. Intuitively, this is similar to a voting process described in the Hough Transform [37]. After all the data elements complete the vote, each voxel will be left with counts, and hence the local histogram, from its neighbors. We call this voting scheme as local deposit and only apply it to all the active voxels.

In our algorithm, we perform the local deposit scheme for each bin in $\mathscr{B}$ separately. While performing the deposit scheme for a single bin, we allocate a *deposit buffer* where each element of the buffer corresponds to a voxel in the dataset. During the deposit procedure, for each active voxel, we locate its neighboring voxels and increment the count in the deposit buffer at each neighboring voxel. After all active voxels for one single bin are processed, the accumulated count at each location in the deposit buffer will essentially be

the frequency for this bin. This procedure for one single bin is explained schematically in Figure 4.2.

4.2.4 Frequency Matching and Collection of Search Results

In this section, we introduce how the deposit count help detect the voxels have similar histogram with the target histogram H_T. During the local deposit stage for a single bin b_k, we calculate the deposit count of bin b_k for each voxel v_i, which is equal to the frequency of bin b_k, $H_{v_i}(b_k)$. By comparing with the target frequency of bin b_k, we can perform the bin-to-bin comparison based on the error metric we select. To determine the final result, V_{final}, we accommodate our algorithm for two types of histogram comparison metrics as follows.

Single-bin-error comparison: For the single-bin-error comparison, all the voxels that match within the error tolerance are labeled as *candidate voxels*. We define the candidate voxels for bin b_k as:

$$v^{b_k}_{candidate} \equiv \{v_i \mid abs(H_{v_i}(b_k) - H_T(b_k)) \leq \delta_{b_k}, \forall i \in \{1, \cdots n\}, b_k \in \mathscr{B}\} \tag{4.5}$$

Here, δ_{b_k} is the user-defined error tolerance for bin b_k. In Fig. 4.2, suppose the target bin frequency is 3 and δ_{b_k} is set to 0, the candidate voxels can be retrieved using Equation 4.5 and are shown in Fig. 4.2 (c). After performing the local deposit and the comparison procedure for all the target bins, the intersection of the candidate voxels from all the target bins are the final result V_{final} (Equation 4.6) which is given as:

$$V_{final} = \bigcap_{b_k \in \mathscr{B}} V^{b_k}_{candidate} \tag{4.6}$$

The step-by-step search algorithm is shown in Algorithm 1.

Algorithm 1 Optimized local historam search algorithm for single-bin-error

Let db_{v_i} be the deposit buffer at voxel v_i and initialize to 0, $db_{NBR(v_i)}$ represents the deposit buffer at neighborhood voxels of v_i including itself. Let E_{v_i} be the error accumulator at voxel v_i and initialize to 0. Let $v_{active}^{b_k}$ be the active voxels for bin b_k
Let $\{b_{k1}, \cdots, b_{kp}\}$ be the sequence of processing bin sorted by $\|v_{active}^{b_k}\|$ in increasing order. $b_k \in \mathscr{B}$, p is the number of bins in $\mathscr{B}$
for $i = 1$ to p **do**
  if $i == 1$ **then**
    $v_{temp} \leftarrow v_{active}^{b_{k1}}$
  else
    $v_{temp} \leftarrow v_{active}^{b_{ki}} \bigcap NBR(v_{candidate}^{b_{k(i-1)}})$
  end if
  for $j = 1$ to $\|v_{temp}\|$ **do**
    $db_{NBR(v_{temp}^i)} \leftarrow db_{NBR(v_{temp}^i)} + 1$
  end for
  for each $v_x \in NBR(v_{temp})$ **do**
    if $abs(H_{v_x}(b_{ki}) - H_T(b_{ki})) \leq \delta$ **then**
      $v_{candidate}^{b_{kj}} \leftarrow v_x$
      $E_{v_x} = E_{v_x} + abs(H_{v_x}(b_{ki}) - H_T(b_{ki}))$
    end if
  end for
end for
return $v_{candidate}^{b_{kp}}$

Sum-of-error comparison: For the sum-of-error metric scheme, we use the same algorithm as the single-bin-error comparison but record the accumulated errors computed from each bin for all the voxels in the search region. Then we compare the final accumulated errors from all the bins in $\mathscr{B}$ with the total error threshold δ_{sum}. For the voxel outside the search region, the accumulated deposit count should be zero since no voxels that belong to $\mathscr{B}$ are in the neighborhood region. In this case, the error can be simply computed by summing the target frequencies in $\mathscr{B}$. By doing so, we can save storage and time for updating errors outside the search region. After collecting deposit counts from all the bins in $\mathscr{B}$,

we scan through all the voxels and check their accumulated errors, and output the V_{final} by filtering out the voxles whose errors are greater than the user-specified error threshold δ_{sum}.

4.2.5 Local Deposit Workload Reduction for Single-bin-error Comparison

If the single-bin-error comparison scheme is selected, we can apply two strategies to reduce the local deposit workload in our algorithm. The first one is called *outcome transition*, in which after collecting and comparing bin frequency for one single bin at each voxel, we pass the resulting candidate voxels to the next bin to shrink the search region. The idea is that if a voxel is rejected in any bin comparison, it will never become a member of V_{final}. As a result, we can skip those voxels and only perform local deposit and frequency comparison at the locations of the candidate voxels for the rest of bins. The second workload reduction strategy is called *ordered processing*. From our observation, the number of outcomes (candidate voxels) for each bin is limited by the number of active voxels in that bin. In other words, the number of candidate voxels will likely be smaller when a smaller number of active voxels exists. Therefore, we first rearrange the processing order for the bins in $\mathscr{B}$ by sorting the number of active voxels associated with each bin in the increasing order and then perform the local deposit for the bins according to the rearranged processing order. By applying ordered processing, the search region can be reduced quickly after the comparing process is performed on the first few bins. Consequently, less workload is needed for the bins that have larger number of active voxels. The numbers of active voxels for all bins are pre-computed and stored along with the bitmaps.

Table 4.1: Three different test datasets in this work. The data size and the bitmap size for each test dataset are also shown.

Dataset	Resolution	Datasize	Bitmapsize	Test Cases
Isabel	$500 \times 500 \times 100$	95.3 MB	35.8 MB	Hurricane Eye
				Ocean
Combustion	$480 \times 720 \times 120$	158 MB	35.1 MB	mixture mass
				Pure fuel
Plume	$504 \times 504 \times 2048$	1930 MB	807 MB	turbulent region
				small velocity region

4.3 Results

In this section, we compare the performance of our method with an existing local histogram search algorithm for both error metrics. Then we show case studies using our local histogram search system.

4.3.1 Performance

The tests were conducted on a machine with Intel Core 2 Duo E6750 CPU, 8GB system memory, and an nVidia GeForce GTX 460 GPU with 1GB of texture memory. Three datasets with different resolutions were used: the pressure field of Hurricane *Isabel* from the Vis'04 Contest; the mixture fraction field of the *Combustion* phenomenon provided by the Sandia National Laboratories; and *Plume* is the Solar Plume simulation for thermal downflow plumes on the surface layer of the Sun, where the scalar field in use was the velocity magnitude, provided by the National Center for Atmospheric Research. The data sizes and the storage sizes of the compressed bitmap indices for all the datasets are listed in Table 4.1. The number of bins used for bitmap index was 256 for all the datasets.

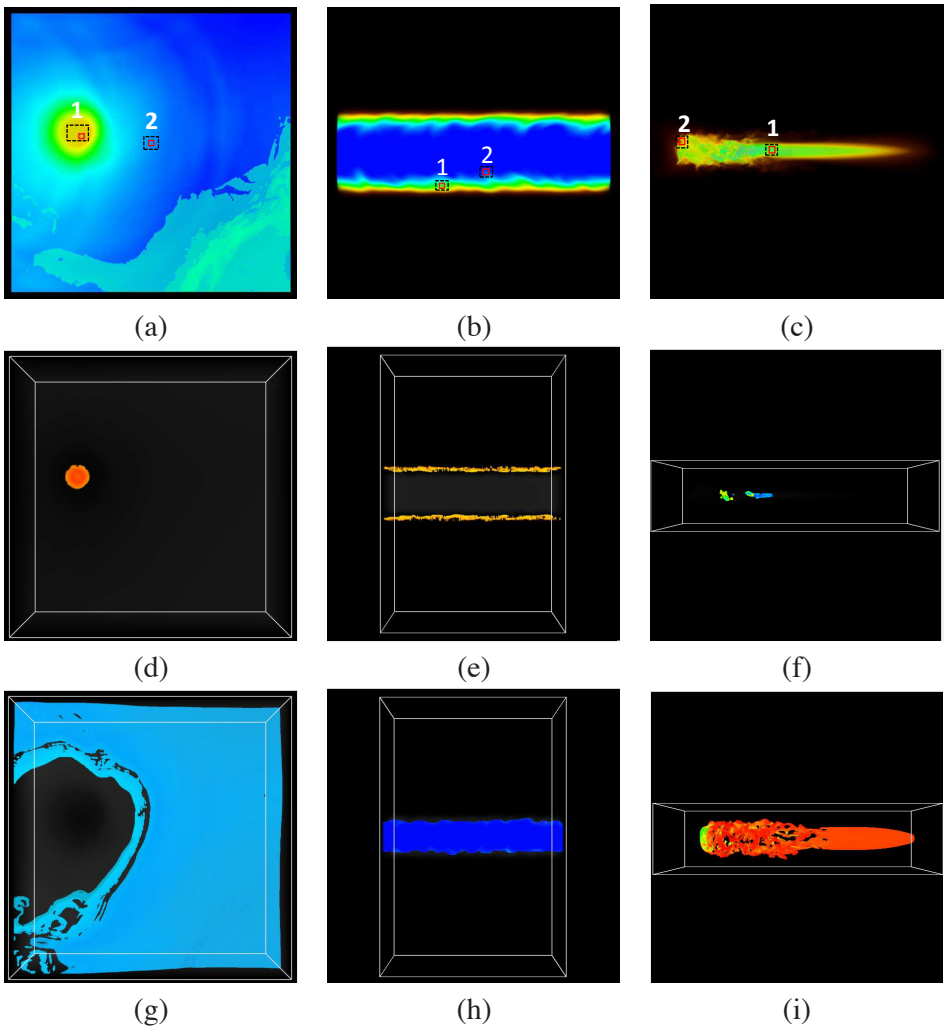


Figure 4.3: The search result images for the Table 4.1. Black boxes in (a)-(c) are the two regions that we used for defining our target histogram. Red box are used for deciding the neighborhood size. (d)-(f) are three cases with small-sized features. (g)-(i) are three cases with large-sized features.

Performance Evaluation and Comparison

To evaluate our algorithm, we compare the performance between our method and the work proposed by Sizintsev et al. [79]. Sizintsev et al.'s method utilizes spatial coherence to

perform more efficient computation time compared to most of the existing local histogram search algorithms. We extended their method of processing 2D images to 3D volume data to fit our requirements in the experiments. To have a fair comparison, we also only consider bins in $\mathscr{B}$ while comparing a local histogram with the target histogram.

Two test cases were conducted in the experiments. In the first case, we tried to match some small-sized features compare to the whole data domain, which include the hurricane eye in the *Isabel* dataset, the mixture mass of fuel and oxidizer in the *combustion* dataset, and the turbulent region in the *Plume* dataset, as shown in the region 1 in Fig. 4.3 (a)-(c). The resulting V_{final} of each dataset are shown in Fig. 4.3 (d)-(f). We also tested our algorithm in one more case for each dataset where the resulting V_{final} contains more voxels compared to the previous case. The second case includes the ocean in the *Isabel* dataset, the pure fuel mass in the *combustion* dataset, and the regions with small velocity in the *Plume* dataset and the resulting V_{final} are shown in Fig. 4.3 (g)-(i). To get a fair performance in each case, we gathered the search computation times with multiple target histograms randomly selected in the region has similar feature (shown as the black box in Fig. 4.3 (a)-(c)), and averaged the collected timings.

The neighborhood size used in the experiments shown in Table 4.1 is 11^3. For the small-size feature cases, the number of bins in $\mathscr{B}$ are 56, 56 and 64 for *Isabel*, *Combustion* and *Plume*, respectively. For the large-size feature cases, the number of bins in $\mathscr{B}$ are 24, 13 and 8, respectively. The features in the second test cases are within small value ranges since they are more likely to be the background, which has few value changes but locates in vast areas. Although the numbers of bins in the second test cases are smaller, the larger number of active voxels results in slower performances than first test cases. Since different

Table 4.2: The performance of the three test datasets in this work.

Dataset	Test Cases	Computation times(s)				Memory Used for Bitmap Index	% voxels Contributed to V_{final}
		Single-Bin-Error		Sum-of-Error			
		Ours	Sizintsev	Ours	Sizintsev		
Isabel	Hurricane Eye	0.273	46.52	0.445	55.45	0.254 MB	0.039%
	Ocean	17.99	51.85	24.40	56.99	26.09 MB	2.2%
Combustion	mixture mass	9.481	75.47	23.39	96.98	7.671 MB	0.18%
	Pure fuel	18.38	92.53	30.25	96.76	33.72 MB	5.45%
Plume	turbulent region	8.051	993.1	29.31	1119.8	11.21 MB	0.016%
	small velocity region	72.42	1257.5	140.91	1240.1	86.41 MB	0.61%

target histograms are selected in each case, the error threshold should also be changed accordingly in order to find meaningful results.

The computation times of single-bin-error and sum-of-error comparison and the total memory usages are listed in Table 4.2. As shown in Table 4.2, for both histogram comparison metrics, the computation times in the cases with large-sized target features are generally higher than that in the cases with small-sized target features. This is mainly because we need to perform the local deposit for more active voxels. We also compare the performance of histogram search using different neighborhood sizes. Fig.4.4 shows the results in different search cases with neighborhood size 11^3, 15^3 and 21^3, as shown in Fig.4.3.

Evaluation of Local Deposit Workload Reduction

To show the impact of the workload reduction scheme introduced in Section 4.2.5 for the single-bin-error comparison, we implemented three different local deposit approaches. The first one includes both workload reduction scheme. The second one only includes outcome transition scheme. The third one excludes both workload reduction schemes. In the experiment, we set the same input target histograms and parameters, such as the neighborhood size (11^3) and the error tolerance (0.001). As shown in Fig. 4.5, the implementation

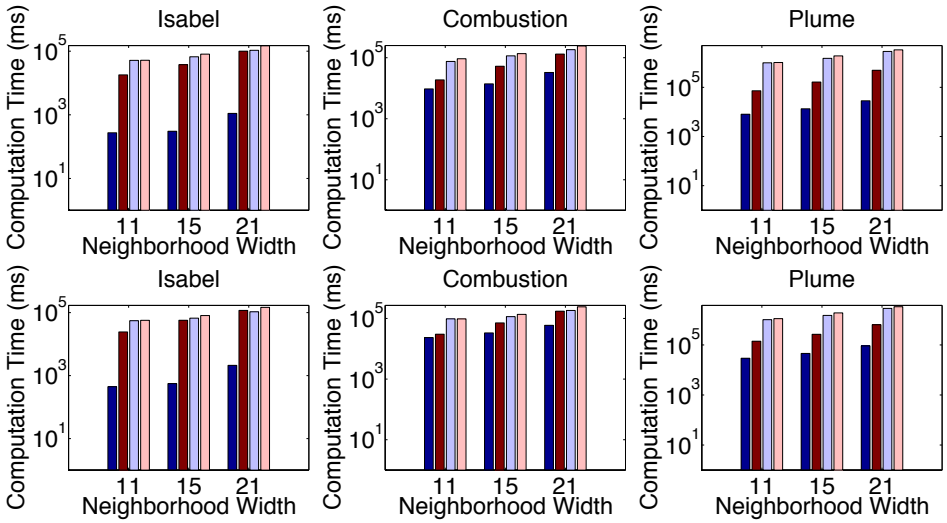


Figure 4.4: The performance comparisons between our work and Sizintsev's method. The figures in the upper row are for the cases with using the single-bin-error metric and those in the bottom row are for the cases with using the sum-of-error metric. Blue colors: Cases of Fig. 4.3 (d)-(f). Red colors: Cases of Fig. 4.3 (g)-(i). Dark color: our work. Light color: Sizintsev's method. Note the logarithmic scale was used for the vertical axes

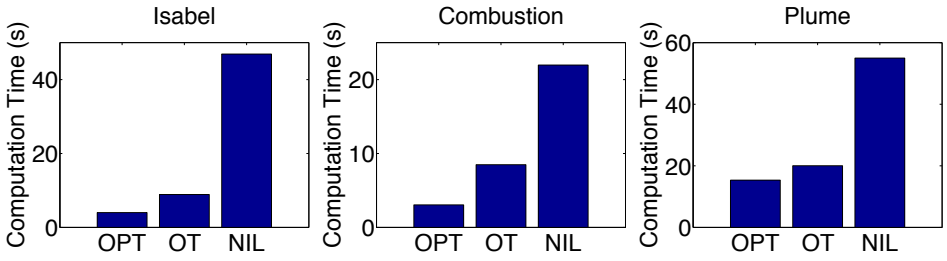


Figure 4.5: Computation time comparison of the cases with different workload reduction schemes for the single-bin-error comparison. The first bar (OPT) shows the test case with using both workload reduction scheme introduced in Section 4.2.5. The second bar (OT) is the test case with outcome transition scheme but without ordered processing scheme. The third bar (NIL) is the test case without both workload reduction schemes .

with both workload reduction schemes performs a much smaller computation time. This shows that the workload reduction scheme is effective for our local histogram search algorithm when the single-bin-error comparison is performed.

Algorithm Parameters Study

In addition to the neighborhood size, the performance of our algorithm is affected by two other factors: the number of bins in $\mathscr{B}$ and the error tolerance. The number of bins in $\mathscr{B}$ is related to the number of active voxels to be processed during the search; the error tolerance is related to the size of the final search result, V_{final}. While these factors may not affect the performance of other approaches that scan through the entire dataset, our method can be affected and gain different speed-up. Therefore, we conducted two experiments to examine these algorithm parameters.

To examine how the total number of active voxels affects our algorithm performance, we tested multiple cases with different number of bins in $\mathscr{B}$. In general, a large number of bins in the target histogram infers a large number of active voxels to process. Therefore, we plot the relations between the computation time and the number of active voxels, as shown in Fig. 4.6. The test result shows that the computation time increases when the total number of active voxels becomes larger due to larger workload of the local deposit.

For the single-bin-error comparison, another important factor of performance is the size of V_{final}, which is related to the number of candidate voxels left after performing frequency comparison for each bin. More specifically, if the user-defined feature is found in vast areas, more voxels have to be visited in the local deposit stage. Consequently, the search space cannot be reduced much by our workload reduction scheme. Images in the upper row of Fig. 4.7 show the increasing trend of the computation time as the number of voxels found increases. For the sum-of-error comparison, we keep all the voxels that have been

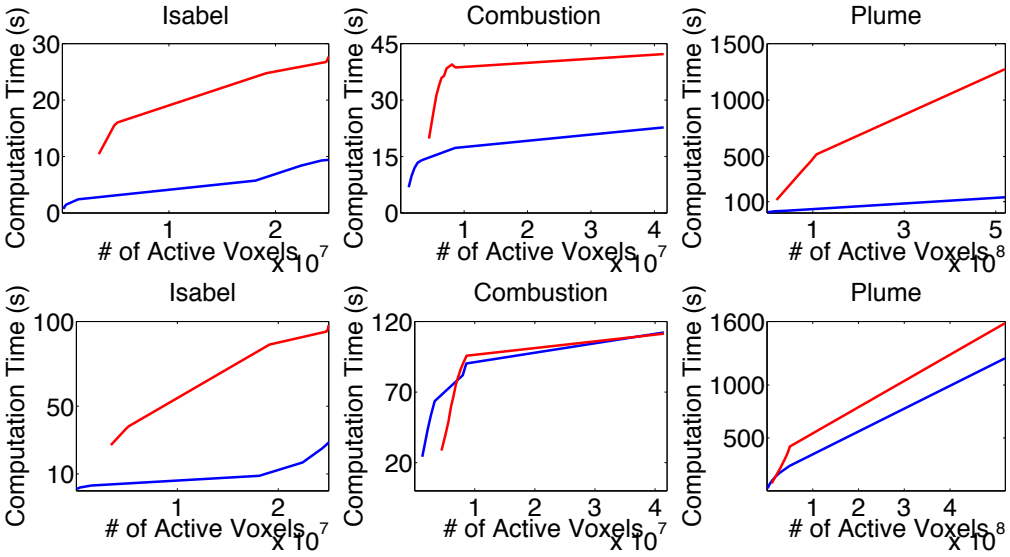


Figure 4.6: Performance comparisons for the cases with having different numbers of active voxels in each dataset. Upper row: Single-bin-error comparison. Bottom row: Sum-of-error comparison. Blue line corresponds to the cases of Fig. 4.3 (d)-(f) and red line corresponds to the cases of Fig. 4.3 (g)-(i).

deposited counts by the active voxels until the end and check the final results in the last stage, so the computation time is not affected by the size of V_{final}, as shown in the bottom row of Fig. 4.7.

4.3.2 Rendering of Fuzzy Local Histogram Search Results

We developed a graphic user interface to assist a user in determining target histograms and also provide the volume rendering of the final result returned by our system. The regions contained the queried features (or V_{final}) are colored by a 1D transfer function that maps scalar values to RGB values. We utilize the similarity value calculated at each voxel in V_{final} to determine the opacity of each voxel. High opacity depicts high certainty in the search result, and low opacity represents high uncertainty. Under this setting, users can

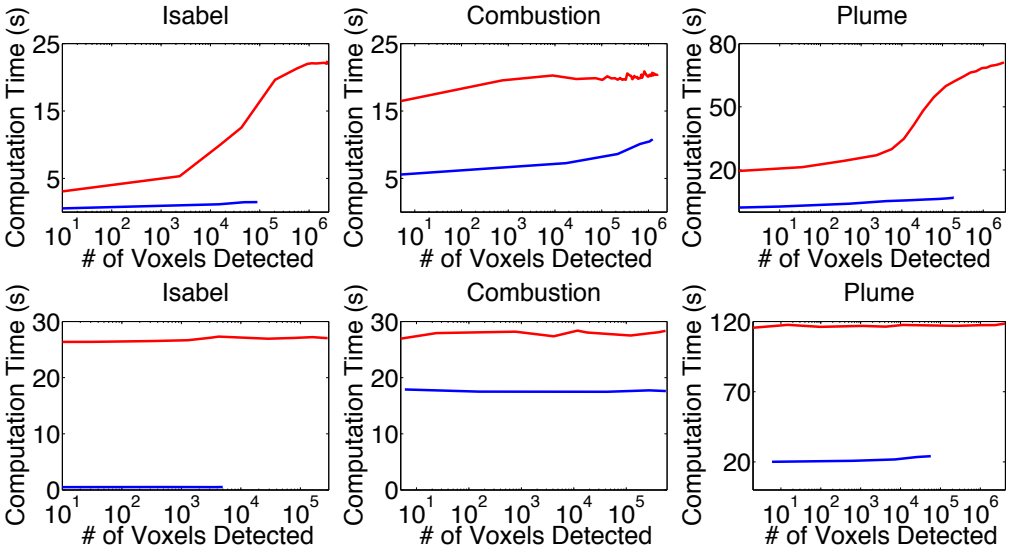


Figure 4.7: Performance comparisons for the cases with having different numbers of detected voxels in each dataset. Upper row: Single-bin-error comparison. Bottom row: Sum-of-error comparison. Blue line: the cases of Fig. 4.3 (d)-(f); Red line: the cases of Fig. 4.3 (g)-(i).

visualize how confidence the region contains the queried feature. The voxels not included in V_{final} are rendered in gray scale to preserve the context of the dataset. To demonstrate the effectiveness of our local histogram search algorithm, we present two case studies from two datasets: *Isabel* and *Combustion* dataset.

Hurricane Dataset

Fig. 4.8 (a) - (e) show the rendering of the search results for the pressure field in the hurricane Isabel dataset. This dataset contains 48 time steps and the grid size for each time step is $500 \times 500 \times 100$. Fig.4.8 (a) shows a volume rendering image of the data in the first time step and we select the region around the hurricane eye as our target features. The red box represents the selected region (9^3 voxels) and the corresponding local histogram is shown on the side of the figure. In this demonstration, we assume the pressure value

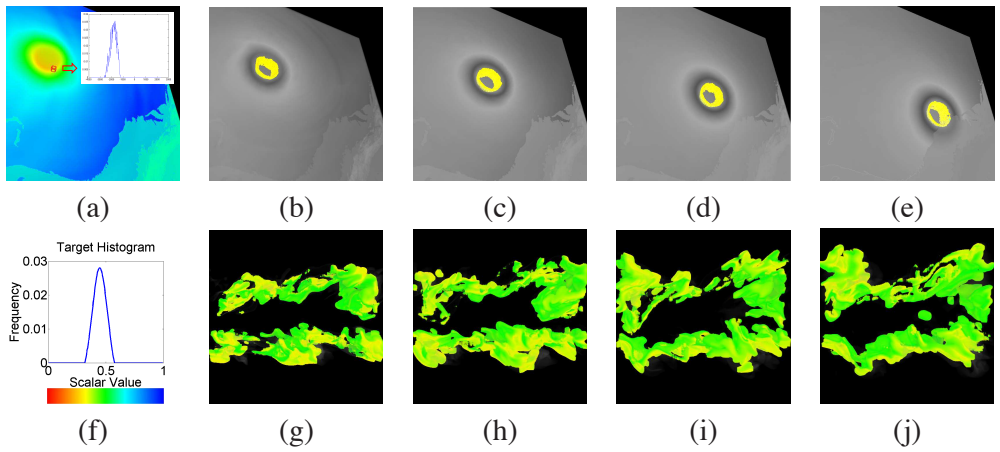


Figure 4.8: Two case studies for demonstrating the effectiveness of our local histogram search algorithm. (a)-(e) are for the Hurricane Isabel dataset. (a): The volume rendering of the dataset at first time step. (b)-(e): the rendering of the search results in time steps 10, 20, 30 and 40, respectively. (f)-(j) are for the *combustion* dataset. (f): the target histogram (g)-(j): the rendering of the search results in time step 60, 70, 80 and 90, respectively.

near the hurricane eye does not vary too much over time and thus, we intend to capture the hurricane eye in the future time steps. The search results for time step 10, 20, 30 and 40 are shown in Fig. 4.8 (b) - (e) and the detected voxels are colored by yellow. The gray regions are shown as the context so that users can visualize the relative position of the hurricane eye and the land. From the rendering results, the hurricane eye is detected in each time step and the movement of the hurricane eye can be captured. Based on the assumption that data values of features do not vary too much in different time steps, our local histogram search algorithm can be applied to feature tracking in the time-varying dataset.

Combustion Dataset

Next, we apply our algorithm to the turbulent combustion simulation dataset. Among the five variables present in the dataset, mixture fraction is used in our experiment. Mixture

fraction is an important variable in the combustion dataset that describes the proportion of fuel and oxidizer mass [2]. In general, the flame can be found at the location with value 0.42. However, due to complex interactions in the fluid, the flame can deviate from this theoretical value.

We describe our target histogram to be a Gaussian-like distribution with the mean at 0.42 and the value range varying between 0.3 and 0.55. The neighborhood size is set to 15^3 in this demonstration. The target histogram is shown in Fig. 4.8 (f) and the detected results in time step 60, 70, 80, 90 are shown in Fig. 4.8 (g)-(j), respectively. From the rendering results, the user can visualize the positional variation of the flame in different time steps.

4.4 Discussion

In this section, we discuss the scope and limitations of our proposed algorithm. In this work, we apply two types of distribution comparison methods: single-bin-error and sum-of-error. In general, the performances for cases with single-bin-error metric are better than those with sum-of-error metric. For the single-bin-error comparison, after comparing frequency for one bin, some voxels will be removed from the search region if their frequencies do not match the corresponding bin frequency in the target histogram. Therefore, search region shrinks quickly and results in higher performance speed-up. For the sum-of-error comparison, we need to keep all voxels in the candidate pool to the end since we have to accumulate all the errors from all the bins in $\mathscr{B}$ for determine the final search result. Therefore we do not achieve as much speed-up in this case. However, as shown in Table 4.2, our algorithm still performs better than Sizintsev et al.'s method in these test cases. Another factor that affects our performance is the number of active voxels in $\mathscr{B}$. From Fig. 4.6, we observe that the positive correlation exists between the computation time and the number

of active voxels increases. Therefore, if the target histogram includes some bins whose number of active voxels are large (usually indicates background values), the computation time will increase and can be slower than Sizintsev et al.'s method. From the observation, this occurs when the number of active voxels is larger than 20% of the total number of data points, which can be seen in the case of the ocean target for Isabel data and the pure fuel target for Combustion data. This fact can be found in the bottom row of Fig. 4.6 and in Table 4.2. From this observation, our work is more suitable for searching for features with small size compared to the whole data domain. To summarize, the limitation of our work is the reduced speed-up while searching for features existed in vast areas.

In addition, our work is suitable for the distribution comparison with bin-to-bin error metrics but currently not applicable for cross-bin metrics such as *Earth Mover's Distance* and *Kolmogorov-Smirnov* distance. In the future, we would like to tackle these limitations to develop a more general algorithm.

4.5 Summary

This Chapter presents an efficient algorithm to search for voxels whose local histograms match the user-defined target histogram. Based on the idea that the value range in the user-defined feature is generally much smaller than the whole data value range, the search space can be reduced rapidly. In this work, we utilize bitmap indexing to quickly locate voxels whose values fall into the user-defined value range to determine the search space and propose the local deposit scheme to determine the frequency at each bin at voxels in the search space. Compared to existing local histogram search algorithms, our method is faster in all the cases applying the single-bin-error comparison and in the cases with a small number of active voxels using sum-of-error metric.

Chapter 5: Efficiently Searching for Features in Multi-field Datasets

5.1 Introduction

Efficient feature search plays an important role in scientific data analysis and visualization, as it greatly reduces visual exploration time and allows users to focus on regions of interest for further analysis and decision making[42]. Several feature descriptors have been proposed for different search applications [22, 56, 76, 90], among which are local statistic-based approaches that define features according to the value distributions derived in local regions [41, 52, 58]. In this work, we describe features as histograms in local neighborhood. Features based on scalar value distributions have been commonly used in query-driven visualization for scientific datasets in recent years [17, 29, 42].

Many scientific datasets are associated with multiple fields, such as pressure and temperature fields in climate datasets. There is a need to define feature descriptors with combined fields since some features cannot be extracted just from a single field. Multi-field feature definitions can be generally classified into two categories [62]: (1) features defined from individual fields separately; (2) features defined using information encoded in a joint set of fields. In our work, we provide efficient algorithms for searching both types of features, based on local distributions, which are defined as marginal features and joint features.

A marginal feature is defined by the marginal distribution in each field, while a joint feature is defined by the joint distribution from a set of joint fields. Compared to marginal features, joint features more precisely describe the occurrence of joint values. Kindlmann and Durkin [48] used joint features with data values and the first and second derivatives to detect boundaries in transfer function design. Pass and Zabih [67] applied joint color distributions to retrieve content-based color images.

When comparing two distributions encoded as histograms, the distance of bin frequencies between two histograms is computed. Then the distances for all bins are accumulated to define the similarity between the two histograms, which is usually defined as bin-by-bin dissimilarity measures. But, local distribution comparisons in large multi-field datasets is difficult due to the high computational cost. There are two issues for computational costs: First, the number of histogram bins increases exponentially as the number of fields or dimensions increases. Second, a large local neighborhood size, which defines the number of neighborhood voxels used for comparing local histograms, can make the search cost high. As a result, the feature neighborhood size was restricted in previous works [42, 58] to avoid a high performance penalty.

In this work, we propose two independent and complementary algorithms for accelerating local distribution searches in multi-field datasets. Both algorithms first approximate a search result, and then use a low-cost refinement step to generate the final search result. The first approach, merged-bin-comparison (MBC), compares multiple histogram bins between two histograms in one pass, instead of comparing individual bins. Utilizing a property of distance measures, our approximate search result from MBC has no false negatives so that the refinement process only needs to remove the false positives to generate the final result. This approach alleviates the performance bottleneck when the target

distribution contains numerous bins. Secondly, we utilize a data sampling method called sampled-active-voxels (SAV). This utilizes stratified sampling to quickly generate approximate initial results, which are close to the final results when compared to simple random sampling. Data sampling increases search performance when searching for large spatial features. Experiments show that both of our improved feature search methods perform much faster than previous methods while retaining a small memory footprint, even when the number of fields in the dataset increases.

The contributions of this paper are:

1. We provide a histogram-based feature search method for multi-field scientific datasets, for both marginal and joint features.

2. We propose an approximate algorithm with low-cost refinement to overcome the curse of dimensionality for multi-field local distribution feature search.

3. We propose a sampling-based approach to achieve efficient large distribution-based feature searches.

5.2 Method

5.2.1 System Overview

We provide an efficient algorithm to search local features which are described by local distributions in a multi-field domain. The framework of our method is illustrated in Figure 5.1. To define a target query feature, a user selects a region of interest and the scalar values in that region are collected to construct the target distribution. In this work, only the bins with non-zero bin frequencies in the target distribution are used for distribution comparison. We provide local distribution search in two types of multi-field features - marginal and joint features. For the marginal feature search, the target distribution is defined accordingly in

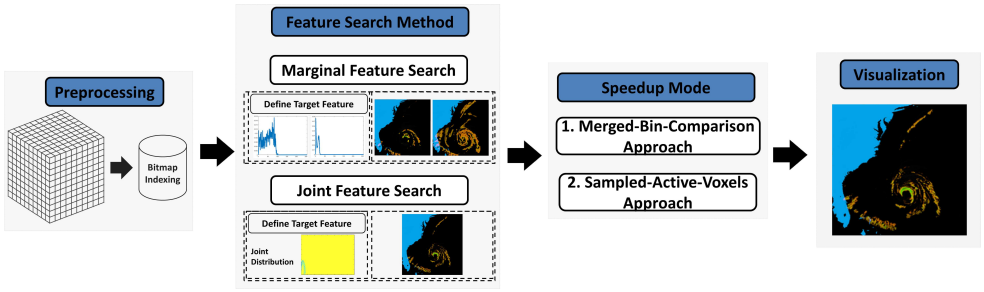


Figure 5.1: The flow chart of our local joint distribution based feature search algorithm.

each field, and the features are searched for independently. For the joint feature search, the local joint distribution is defined and extracted from a joint set of fields. We search for the joint features by comparing local joint distributions to the target joint distribution.

In both cases, we use bitmap index to search for distribution based features. In Chapter 4, we proposed the local deposit method to efficiently achieve local distribution search for a single field. However, the performance issues for multi-field domain search are still not completely addressed. First, the number of histogram bins to process during distribution computation and comparison is greatly increased for multi-field datasets. This increases the search workload during joint feature search. Second, large spatial features (local neighborhood search sizes) result in expensive distribution comparisons. We propose two strategies to efficiently search local distributions in multi-field datasets. The first is to reduce the number of bin frequencies comparison by merging multiple bins comparisons into one single process. Furthermore, the performance of the local deposit method is primarily dependent on the number of active voxels, so we reduce the number of active voxels by applying stratified sampling. For both methods, refinement is applied to produce the final search results. Next, we first introduce the two types of features, marginal and joint, in local distribution searches and present our acceleration approaches.

5.2.2 Marginal Feature Search for Multi-Field Datasets

For the marginal feature search, a query generates a set of *resulting voxels* $\mathscr{V}$ whose local distributions match their respective target distributions for each field. Given d fields, boolean operators are applied to all sets of resulting voxels $\mathscr{V}_1, \ldots, \mathscr{V}_d$, similar to [96], assembling multiple search predicates into one result.

AND: $\mathscr{V}_1 \wedge \cdots \wedge \mathscr{V}_d$ yields a set of voxels where the local marginal distributions match the target distributions for all fields. This operator provides the user to search and visualize only the regions where all the distribution features co-occur, which is also referred to as the intersection of multiple distribution based features.

OR: $\mathscr{V}_1 \vee \cdots \vee \mathscr{V}_d$ yields voxels that at least one of the local distributions matches its respective target distribution. For this operator, it allows the user to search and visualize all the regions that include at least one of target features defined in the selected fields, which is also referred to as the union of multiple distribution-based features.

We utilize a bit vector to mark the resulting matched voxels for each field in the bitmap. Boolean operations across all fields can be applied to generate the combined set of resulting voxels. Furthermore, if the *AND* operator is applied, we can use the resulting voxels from a query as the input search region for the next field query, optimizing the efficiency of our search.

5.2.3 Joint Feature Search for Multi-Field Datasets

To efficiently search for joint features, we also leverage bitmap index, which is generated in a pre-processing step. When a user selects a feature region of interest, a new bitmap for modeling a set of joint fields will be constructed from the bitmaps of all fields. This

Table 5.1: An example of the joint distribution based bitmap.

Dataset			Bitmap Index							
Voxel ID	Data Value		b_0		b_1		b_2		b_3	
	variable 1	variable 2	[0,1)	[10,20)	[1,2)	[20,30)	[2,3)	[30,40)	[3,4]	[40,50]
0	0.15	12.34	1		0		0		0	
1	1.23	22.54	0		1		0		0	
2	2.16	35.21	0		0		1		0	
3	2.98	38.55	0		0		1		0	

new bitmap contains bit vectors for the non-zero frequency count bins of the local joint distribution, defined as $\hat{\mathscr{B}}$ in multi-field cases.

Each bin in this new joint distribution represents one combination of value ranges across multiple fields, $(R_1, \cdots, R_d)$, where d represents the number of fields. A combination of value ranges (a bin of the joint distribution) associates a bit vector which encodes which voxels' value vectors of multiple fields fall into that combination of value ranges (in=1, out=0). To generate the bit vector for one combination of value ranges (called bin q), we apply *AND* operations on all of the bit vectors representing the respective value ranges in all user-selected fields, which is formulated as follows:

$$bmp_{JD}(bin_q) = bmp_1(R_{1,q}) \wedge bmp_2(R_{2,q}) \cdots \wedge bmp_d(R_{d,q}) \tag{5.1}$$

Here, $bmp_{JD}(bin_q)$ represents the bit vector for bin_q, where the joint distribution associated with a combination of value ranges $R_{1,q}, R_{2,q}, \cdots R_{d,q}$ are defined for each field. An example of a bmp_{JD} is shown in Table 5.1.

To apply the local deposit method for joint feature search, the active voxels defined in Equation 4.4 can be re-defined for multi-field cases as

$$V'_{active} \equiv \{v_i \mid \vec{f}(v_i) \in \text{combinations of user-defined value ranges from all fields}, \forall i \in \{1, \cdots, n\}\} \quad (5.2)$$

Here, $\vec{f}(v_i) = \{f_1(v_i), f_2(v_i), \cdots f_d(v_i)\}$. $f_d(v_i)$ is the corresponding value at voxel v_i in the d_{th} scalar field, n is the total number of voxels. Notice that the maximum number of voxels in V'_{active} is the total number of voxels in the dataset so that it is invariant no matter how many fields are used.

5.2.4 Efficient Search with a Merged-Bin Comparison

The Merged-Bin-Comparison Algorithm

To improve performance when there are many bins in the user-defined value ranges, we propose an efficient search algorithm called the merged-bin-comparison algorithm. From observation, query performance is affected by the number of steps required to compare bin frequencies for all of the bins in $\hat{\mathscr{B}}$. We use the previously described local deposit method to count the value frequencies and compute the distance to each of the target bin frequencies in a search region to locate the candidate voxels. We improve the performance of the local deposit search method by reducing the number of bin comparisons through grouping bins and comparing groups rather than individual bins.

Per individual bin, the L_1-norm is used as the distance metric which compares the target query distribution H_T to a local distribution LH_{v_p} at a voxel v_p, i.e.,

$$D(H_T, LH_{v_p}) = \sum_{bin_k \in \hat{\mathscr{B}}} \left| H_T(bin_k) - LH_{v_p}(bin_k) \right| \quad (5.3)$$

$H_T(bin_k)$ represents the frequency value (count) of bin_k in H_T.

From the *triangle inequality* property of absolute values, all real values satisfy the inequality $|a|+|b| \geq |a+b|, \forall a,b \in \mathbb{R}$. Based on this property, it satisfies that $\sum |x_i| \geq |\sum x_i|, \forall x_i \in \mathbb{R}$. Therefore, given any subgroup of $\hat{\mathscr{B}}$ bins (defined as $\hat{\mathscr{B}}'$ where $bin_k \in \hat{\mathscr{B}}'$), the partial distance measure computation using Equation 5.3 satisfies the following inequality:

$$\sum_{bin_k} \left| H_T(bin_k) - LH_{v_p}(bin_k) \right| \geq |\sum_{bin_k} H_T(bin_k) - \sum_{bin_k} LH_{v_p}(bin_k)| \tag{5.4}$$

We define the left-hand side of Equation 5.4 as D and right-hand side as D', hereafter. In D', we observe that $\sum_{bin_k} H_T(bin_k)$ is the sum of several bin frequencies in H_T, which is a fixed value and can be computed as long as the target query distribution is determined. $\sum_{bin_k} LH_{v_p}(bin_k)$ in D' represents the accumulated frequencies of several bins in LH_{v_p}. Instead of calculating frequency differences bin-by-bin using D, we calculate them in aggregate once, using D', by combining multiple local deposits into one operation. This is done by merging the active voxels' bit vectors into one bit vector through a bit-wise *OR* operation. Then, we perform local deposit for this set of active voxels to calculate their accumulated frequencies $\sum_{bin_k} LH_{v_p}(bin_k)$. Therefore, D' can be computed by this one local deposit and frequency comparison. Comparatively, if we were to compute the left-hand side of Equation 5.4, it would be very time-consuming due to many calculations of frequency differences.

We take the advantage of this fast computation of D' to be able to compare the target distribution and local distributions with generating approximate search results. To do so, we partition $\hat{\mathscr{B}}$ (to get $\hat{\mathscr{B}}'$) bins into m groups. From Equation 5.4, D' is less than or equal to D for each group in the m groups of bins. In other words, the difference (distance) between two distributions calculated by D' will always be less than or equal to the difference calculated by D. Given a match tolerance of δ, the generated search result by applying

D' (an approximate result, $\mathscr{V}_{app}$) is always the superset of the result generated by D (final search result, $\mathscr{V}_{final}$):

$$\mathscr{V}_{app} \equiv \{v_p \mid \sum_{g=1}^{m} D_g^{'}(H_T, LH_{v_p}) \leq \delta\} \tag{5.5}$$

$$\mathscr{V}_{final} \equiv \{v_p \mid \sum_{g=1}^{m} D_g(H_T, LH_{v_p}) \leq \delta\} \tag{5.6}$$

To get $\mathscr{V}_{final}$ from $\mathscr{V}_{app}$, we filter out false positives in $\mathscr{V}_{app}$, which are voxels whose actual distances are greater than the match tolerance δ threshold.

Group Selection for Merged-bin-comparison Algorithm

In MBC algorithm, we need to determine the number of groups and which bins are joined into one group to minimize the error between D' and D. With a smaller error, the approximate search result will be closer to the final search result, which implies smaller computational overhead to refine the approximate results. From observation, if the signs for all $H_T(bin_k) - LH_{v_p}(bin_k)$ (denoted as φ_{bin_k} hereafter) in a group are similar, the error between D' and D will be small. For example, the error will be zero when all of the signs of φ_{bin_k} are simultaneously all positive or all negative. Furthermore, we also observe that the signs of φ_{bin_k} are more consistent when bins that have similar target frequencies are grouped together.

Therefore to minimize errors between D' and D, we empirically group bins by similar target bin frequencies. Furthermore, if a bin group contains too many bins, it will increase the chance to have higher error between D' and D. To avoid this, we split the initial groups that contain too many bins into subgroups. Equation 5.7 demonstrates the reduction of error between D' and D by increasing the number of groups.

$$\sum_{k=1}^{i} \left|\varphi_{bin_k}\right| \geq \left|\sum_{k=1}^{j} \varphi_{bin_k}\right| + \left|\sum_{k=j+1}^{i} \varphi_{bin_k}\right| \geq \left|\sum_{k=1}^{i} \varphi_{bin_k}\right| \tag{5.7}$$

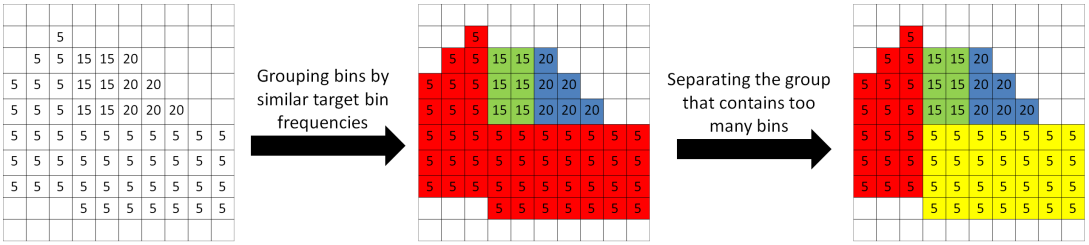


Figure 5.2: An example of group selection for MBC algorithm. All grids represent the 2D joint histogram of a target feature. The number in each cell is the value frequency for that bin. Bins of the same color have been grouped together. The left figure shows the joint histogram before grouping. The middle figure represents grouping bins by target frequency. The right figure represents separating the red group into two groups by bin indices.

The leftmost term represents D, which is the true frequency difference of bins in $\hat{\mathscr{B}}'$ (a subgroup of $\hat{\mathscr{B}}$). The rightmost term represents D' with one group of bins, while the middle term represents splitting D' (splitting $\hat{\mathscr{B}}'$ into two groups of bins), reducing the error between D' and D. To split a group of bins into several subgroups, we use K-means [43] to cluster the bins by their indices. To balance the performance and accuracy, we only look for groups that contain more than 100 bins, and split them into 10 clusters each. After regrouping, a new bit vector is built for the two new groups, performing *OR* bit vector operations for each group.

Figure 5.2 shows an example of group selection for histogram bins. Initial group selection is determined by bin frequencies, as shown in the middle figure. Since the red group contains too many bins, it is split further into two groups, the red and yellow groups, as shown in the right figure.

5.2.5 Efficient Search with a Sampled-Active-Voxels Method

The neighborhood size is one of the key factors for the computational cost of constructing a local histogram. In previous work [42, 56, 58], they either restricted the neighborhood size or randomly selected samples in the neighborhood region for a voxel. We opt for a sampling strategy to avoid large computation overheads. However, there is an issue with vanilla random sampling for generating local histograms, which is that the distribution characteristics of local histograms may not be well preserved after sampling.

To address this issue, we apply the *stratified sampling* method proposed by Su et al. [81]. Compared to random sampling, searching local distributions using stratified sampled data can generate approximate results closer to the ground truth. Figure 5.3b shows empirical evidence, displaying the distribution of errors between approximate results and the ground truth. These errors represent the differences between local histograms constructed from a down-sampled dataset and the full dataset. From our experiments, the errors for the stratified sampling case (black histogram) are closer to zero when compared to random sampling case (red histogram).

Stratified Sampling

We apply the stratified sampling to reduce the workload of performing local deposit and counting bin frequencies by reducing the number of active voxels for each bin of the joint distribution. It leverages bitmap indexing to distribute data samples across a scalar value distribution. While constructing the bitmap indices, we have partitioned the data value range into several bins (strata) for each field. A bit vector is created for each these value ranges (one bit vector for one bin in the joint distribution), as described in Section 5.2.3. To perform sampling, we select an equal percentage of samples, per value range, by creating

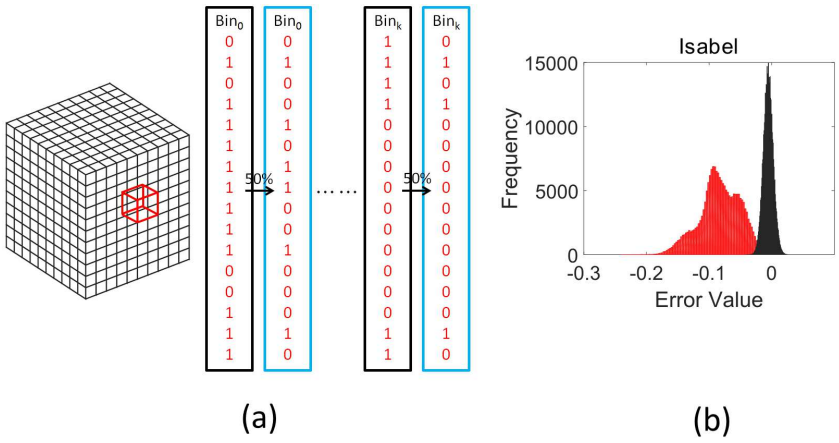


Figure 5.3: (a) An example of stratified sampling within a cubical region. The sampling percentage is 50% in this case. (b) The distribution of errors between approximate results and ground truth for the *Isabel* dataset. The black one is the case of stratified sampling and the red one is the case of random sampling.

a new bit vector with the bits randomly turned on if they are already on. This procedure using bit vectors to sample the data is equivalent to *stratified random sampling*. The value distribution in a resulting sampled dataset, by distributing samples across the value range bins, is closer to the original dataset when compared to simple random sampling.

To increase the accuracy of sample value distributions from a spatial region, we subdivide each bit vector into partial bit vectors based on a spatial block decomposition. The reason for this is that we define the local spatial neighborhood of a voxel as cubical blocks. Random samples are then allocated to these spatial-value range bit vectors. Figure 5.3a shows an example of stratified sampling in combined spatial-value regions. The bit vectors here are only for one local spatial region (the red block region in the cube). The black rectangles represent bit vectors for particular value ranges, and the blue rectangles represent those bit vectors after 50% sampling. To allocate samples to that spatial block, we

randomly turn off 50% of the bits. We repeat this process for each spatial block, in each value range bit vector. In this paper, we used a block size of 7^3 when performing the spatial subdivision of value range bit vectors.

Error Estimation

We attempt to preserve the distribution within a local cubical blocks through stratified sampling, but the distribution will still have differences from the local distribution in the full, unsampled data. Therefore, we provide an error estimation method, using pre-processed data, to calculate the difference (error) between an approximate search result and the true search result using the full dataset. This estimate is used to tune the error tolerance during search when using a sampled bitmap, and it reduce false negatives with a given confidence interval.

The difference between an approximate and true search result, given a sampling percentage s, is formulated as follows:

$$\mathscr{E}_s = \sum_{k=1}^{i} (|H_T(bin_k) - LH(bin_k)| - |H_T(bin_k) - LH_s(bin_k)|) \tag{5.8}$$

LH_s represents the local distribution constructed from the sampled data with the percentage s. From Equation 5.8, $\mathscr{E}_s$ can only be computed by knowing the target distribution H_T, and it follows that it can only be computed during the search procedure. Instead, we apply an inequality theorem to generalize the computation of $\mathscr{E}_s$ for different inputs of target distributions H_T, such that $\mathscr{E}_s$ can be estimated in a pre-processing stage and used prior to searching.

First, all positive values satisfy the inequality $a - b \leq |a - b|, \forall a, b \in \mathbb{R}$ and $a, b \geq 0$. Therefore,

$$\mathscr{E}_s \leq \sum_{k=1}^{i} \left|\left(|H_T(bin_k) - LH(bin_k)| - \left|H_T(bin_k) - LH_s(bin_k)\right|\right)\right| \tag{5.9}$$

Second, from the *reverse triangle inequality* property of absolute values [45], all real values satisfy the inequality $|(|a|-|b|)| \leq |a-b|, \forall a,b \in \mathbb{R}$. Based on this property, it satisfies that $\sum |(|x_i|-|y_i|)| \leq \sum |x_i - y_i|, \forall x_i, y_i \in \mathbb{R}$. Therefore,

$$\mathscr{E}_s \leq \sum_{k=1}^{i} |LH(bin_k) - LH_s(bin_k)| \tag{5.10}$$

The right-hand side of Equation 5.10 (denote it as $\mathscr{E}_s^*$ hereafter) can be computed from the local distribution constructed from the original dataset and the sampled dataset.

However to calculate this, it would take too much computational time and storage for $\mathscr{E}_s^*$ for all voxels. To avoid computing $\mathscr{E}_s^*$, we apply the *central limit theorem* (CLT) to estimate the mean and the standard deviation of all $\mathscr{E}_s^*$ values. CLT is a well-known approach to efficiently approximate the mean and standard deviation for most distributions. The steps of CLT are:

1. Randomly select n values from the random variable and calculate the mean of these n values. In our work, the $\mathscr{E}_s^*$ values, for all voxels, is our random variable.

2. Iteratively process the previous step 1 m times, and calculate the mean $\overline{\mu}$ and the standard deviation $\overline{\sigma}$ for the distribution of these m mean values, which is considered as a *Gaussian distribution*.

3. The mean μ_s and the standard deviation σ_s of the random variable can be approximated as $\overline{\mu}$ and $\overline{\sigma} \times \sqrt{n}$, respectively.

We then use the μ_s and the σ_s to estimate the dispersion of $\mathscr{E}_s^*$ values by applying *Chebyshev's inequality*. Chebyshev's inequality states that there will be no less than $1 - 1/p^2$ of values within p standard deviations of the mean. For instance given a confidence interval 90%, p is 2.236.

We now know the distribution of $\mathscr{E}_s^*$ values and can apply it towards tuning the error tolerance δ, which yields the approximated search results with a given true positive rate

formulated as

$$\delta_{tuned}(p) = \delta + \mu_s + p \times \sigma_s \tag{5.11}$$

5.2.6 Implementation

Our algorithm requires a deposit buffer and an error buffer to store the intermediate results. The deposit buffer is used to store the counts while performing local deposit for each bin (or a group of bins), and the error buffer is used to store the accumulated distances of bin frequencies between local distributions and the target distribution. A naive way is to create the buffers the same size as the dataset to store the counts and accumulated distances. However, this approach is not practical for large datasets. Instead, we partition the data space into multiple non-overlapping spatial blocks and perform local deposit and frequency checking block by block. By applying this region-based approach, we only have to keep two block sized buffers.

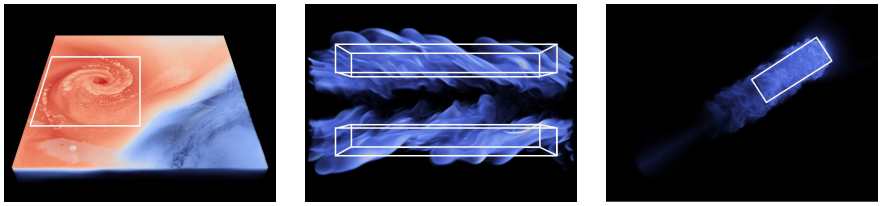
This also allows us to introduce an early termination scheme to further enhance the performance. If the accumulated error for a voxel is greater than the error tolerance, the rest of bin frequency checking can be skipped. Furthermore, if all the voxels in the currently processed region satisfy the skip condition, we can stop local deposit for this region. The *reorder scheme* introduced in Section 4.2.5 is also used to improve the early termination scheme. We rearrange the order of bins in $\hat{\mathscr{B}}$ by sorting the target frequencies in decreasing order, and then perform frequency comparison in this sorted order. With the reorder scheme, we can exclude voxels whose local histograms have great difference in the bins with larger target frequencies as early as possible. These acceleration approaches, along with refinement process implemented in *OpenMP*, greatly increases our overall search performance.

Table 5.2: The fields of three datasets used in the experiments.

Dataset	Resolution	Data Size Per Field	Fields	Bitmap Size
Isabel	1000*1000*200	762 MB	Pressure	170 MB
			QVapor	281 MB
			Temperature	182 MB
Combustion	960*580*240	509 MB	χ	262 MB
			MixFrac	722 MB
			Y_OH	626 MB
Plume	504*504*2040	1930 MB	Curvature	247 MB
			Divergence	248 MB
			Magnitude	807 MB

5.3 Results

In this section, we show the performance of the proposed local distribution search algorithm. Three datasets were used in the experiments: The *Isabel* dataset represents Hurricane Isabel from IEEE Vis'04 Contest, where *Pressure*, *Water Vapor* and *Temperature* fields were used. The *Combustion* dataset is a simulation of combustion phenomena provided by the Sandia National Laboratories, where χ, *Mixture Fraction (MixFrac)* and *Hydroxyl Radical (Y_OH)* fields were used for testing. The *Plume* dataset is a Solar Plume simulation for thermal downflow plumes on the surface layer of the Sun, provided by the National Center for Atmospheric Research. For our studies, we converted the vector fields into three scalar fields, *curvature*, *divergence* and *magnitude*. For each dataset shown in Table 5.2, we randomly selected target local distributions to search for from the highlighted regions in Figure 5.4.



(a) (b) (c)

Figure 5.4: The selected regions for defining the target joint distribution. (a) *Isabel* (b) *Combustion* (c) *Plume* dataset.

5.3.1 Performance

To compare the performance, we randomly selected voxels as search targets from the white boxed area shown in Figure 5.4. For each target voxel, we constructed the local distribution around the voxel as the target feature for distribution search. The number of value bins used in the construction of the local distribution was fixed to 256 per field. The error tolerance δ for the search match distance was set to 0.5, using L_1-norm as the distance measure. All the experiments were tested on a machine with an Intel Core i7-4770 CPU and 16GB of system memory.

Performance Evaluation of Merged-Bin-Comparison

We compare the performance of our MBC algorithm proposed in Section 5.2.4 to the approach without MBC optimizations. In this experiment, we randomly selected 100 features as described previously. The neighborhood size of a local distribution for this test was defined as 11^3. The performance for the three datasets is shown in Figure 5.5. We only compare the performance for joint feature search, since a marginal feature search can be regarded as performing multiple searches with a single field.

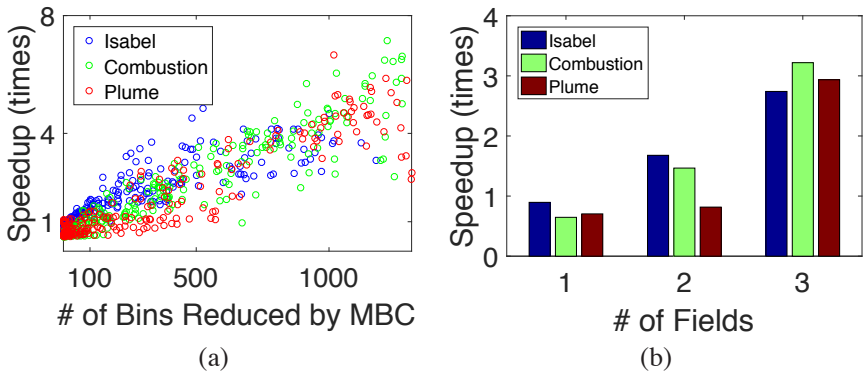


Figure 5.5: The performance speedup gained by our MBC approach compared to the approach without MBC. (a) The individual performance speed-up for all test cases, where the x-axis is the number of bins reduced by MBC. (b) Grouping the test cases by dataset and by the number of fields searched, showing the average performance gain.

Figure 5.5 shows the performance speedup gained from our MBC approach. Figure 5.5a shows all experiments (including single-field and multi-field cases) for the three datasets, where the x-axis is the number of bins reduced by MBC. We can see that as the number of reduced bins increases, so does the MBC performance. We observe that there is almost no speedup in the cases when the number of reduced bins is smaller than 100. This is because, in those cases, fewer reductions of bins results in fewer reductions in the iterations of the algorithm.

Furthermore, we grouped all the tests by their corresponding dataset and the number of fields used to show the average performance gain in Figure 5.5b. Most lower performance gains come from the single-field test cases, while the average 2+ multi-field cases have a larger speed up, i.e., the speed up is larger when the number of fields searched increases. In general, fewer bins are used in a single-field distribution test case, so the effectiveness of our MBC method is limited due to fewer bin merges. Evidence for this is shown in Table

Table 5.3: The numbers of search results for MBC.

Dataset	Isabel			Combustion			Plume		
# of fields	1	2	3	1	2	3	1	2	3
# of bins reduced by MBC	0	174	446	16	305	675	8	203	608
# of grouped bins after MBC	9	27	28	16	28	24	18	25	25
V_{app}	1738668	59592	1383	10214327	266172	9679	4637837	729162	2652
V_{final}	1722512	2733	252	9657854	125267	5819	4229405	68437	697
# of true resulting voxels	1722512	2733	252	9657854	125267	5819	4229405	68437	697

5.3, showing that there are less bin reductions in single-field cases. Therefore, the MBC method is optimal when there are many bin reductions to increase the search performance.

To further demonstrate the efficiency of our work, we compare our performance with Sizintsev et al.'s work [79]. We extended their method to process 3D spatial volume data with higher dimensional joint distributions, showing the performance comparison results in Figure 5.6. The performance complexity of Sizintsev et al.'s work is only affected by the data size and the number of bins of a histogram, and consequently, we only selected one feature from our 100 test cases during this comparison. We selected the one target feature that has a similar number of bins as compared to the average number of bins for all 100 target features, to make a fair comparison between our method and theirs. As shown, we have higher overall performance when compared to [79] in the multi-field cases. For all the three datasets, the computation times for the cases with three fields using Sizintsev et al.'s search method actually took more than one day, and would have been off the chart.

Moreover, two additional points can be observed from Table 5.3. First, the number of grouped bins after applying MBC method ranges between 10 to 30 bins in this experiment, which is independent of the number of fields. Meanwhile, the number of bins in $\hat{\mathscr{B}}$ usually grows rapidly when the number of fields increases. This indicates that the differences between the number of grouped bins and the number of bins in $\hat{\mathscr{B}}$ is large. This means that

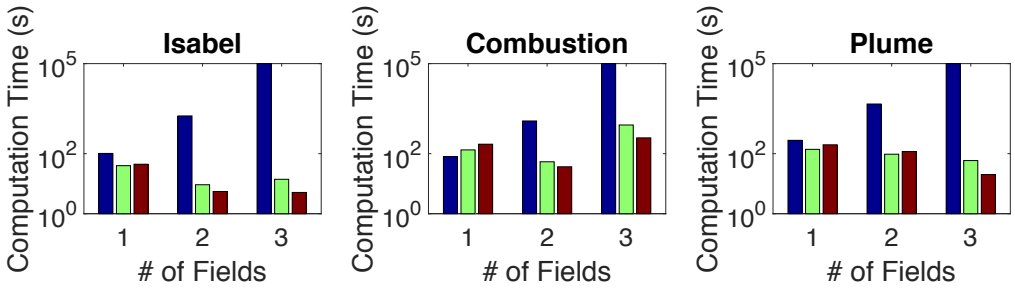


Figure 5.6: The performance comparisons with previous work. Blue bars: Sizintsev et al.'s approach. Green bars: the approach without MBC. Red bars: our MBC method. Notice that the logarithmic scale is used in the y-axes. If computation took more than a day, it is shown as 10^5 seconds on this chart.

our method gains greater speed ups when the number of fields is large and the number of bins grows. Second, due to the effectiveness of our group selection method in MBC, the number of candidate voxels in the approximate result (V_{app}) versus the final search result (V_{final}) do not differ much. This means that we do not have to do much additional work pruning the false positives from the approximate result.

Performance Evaluation of Sampled-Active-Voxels

We show the performance of our SAV method proposed in Section 5.2.5. In this experiment, we randomly selected 20 features for each dataset from the white boxed areas shown in Figure 5.4. The search neighborhood size was defined as 31^3 and the sampling percentage was set to 20%. The value of p, for tuning the error tolerance in Equation 5.11, was 1.414, and the associated confidence interval is 50%. The speedup performance gained by our SAV approach as compared to the method without SAV is shown in Figure 5.7. Like before, we have only compared the performance gain for joint feature search in this experiment.

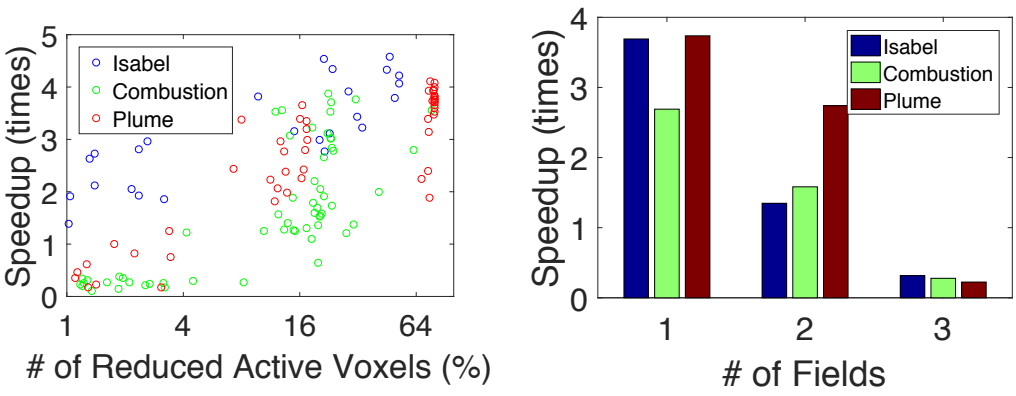


Figure 5.7: The performance speedup gained by our SAV method using 20% sampling, compared to the approach without SAV for joint feature search. (a) All test cases where the x-axis is the percentage of total data points of the corresponding dataset. Notice the logarithmic scale for the x-axis. (b) Grouping the test cases in (a) by data set and the number of fields, showing the average performance.

Figure 5.7a shows the speedup gain from the reduction of active voxels for each feature. Instead of using actual number of reduced active voxels, we convert the number to the percentage of total data points and use the percentage on the x-axis. As we can see, our SAV method performance increases when the number of reduced active voxels increases.

As before, we group the test cases by their corresponding dataset and the number of fields to show the average performance across the groups in Figure 5.7b. We are able to increase performance by about 2.5 to 3.5 times using the SAV method in single-field cases. However, this speedup is reduced in the multi-field cases. From our experiments in these test cases, we observed that larger errors occur between the local joint distributions constructed by full sets and their sampled active voxels versions where there are many bins. This is due to the error generated by $\mathscr{E}_s^*$ in Equation 5.10 such that sampling is accumulated bin by bin. Therefore, the error is large when many bins are included, and it

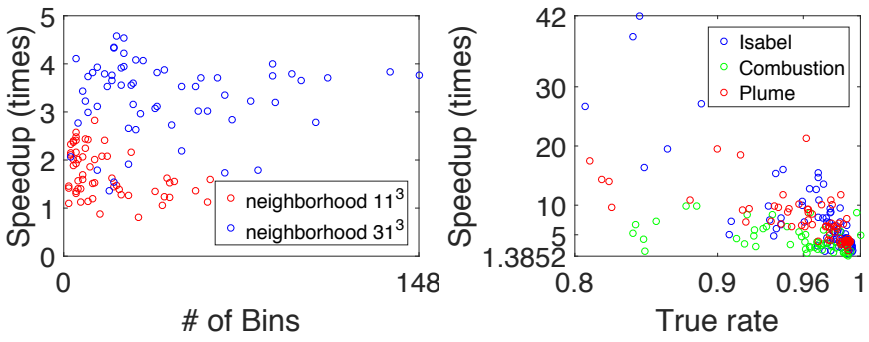


Figure 5.8: The performance comparison of SAV. (a) The comparison between two cases with using different neighborhood sizes. (b) The performance speedup versus accuracy of results.

results in a large amount of false positives in the approximate result V_{app}. This increases the computation time during which false positives are removed to generate V_{final}. This limits the effectiveness of SAV in many multi-field cases, but it achieves a huge performance speedup for the single-field cases, counteracting the lack of performance improvement for MBC in single-field cases.

We compared the performance of SAV using different neighborhood sizes for constructing the local histograms, shown in Figure 5.8a. In this experiment, we applied marginal feature search for each of the fields in three datasets using two different neighborhood sizes, 11^3 and 31^3. The performance speedup, for 20 features due to the SAV approach, is approximately 3 to 5 times when the neighborhood size is 31^3. This is much better than neighborhood sizes of 11^3, which only achieves approximately 1.2 to 2.5 times speedup. Consequently, SAV is more optimal for increasing the performance for large feature search.

Finally, we compared the computation times of joint feature search with the Sizintsev et al.'s approach, the approach without MBC and SAV, our MBC, and our SAV, shown in Figure 5.9. In all cases, our MBC method (yellow bars) or our SAV method (red bars)

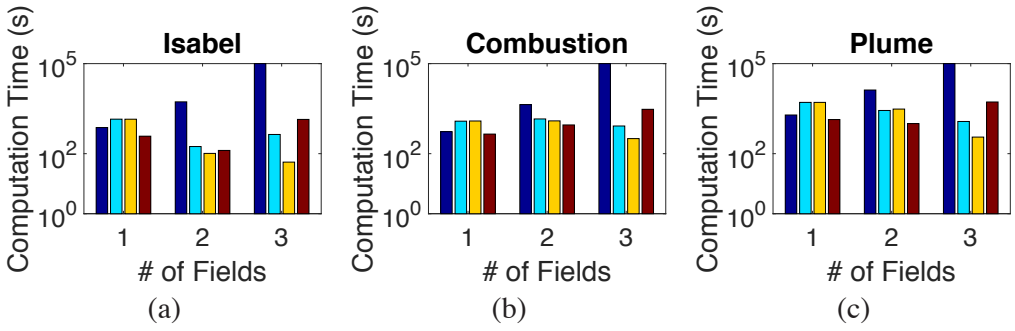


Figure 5.9: The computation time comparisons for joint feature searches with neighborhood size 31^3. Dark blue bar: Sizintsev et al.'s approach. Light blue bar: The method without MBC and SAV. Yellow bar: MBC. Red bar: SAV. Notice the logarithmic scale on the y-axes.

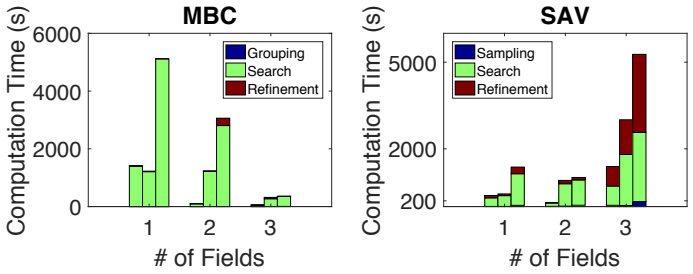


Figure 5.10: The computation time spent in different phases of the MBC and SAV approach, grouped by number of fields in the test. The three bars per group represent the Isabel, Combustion and Plume datasets (from left to right), respectively.

performs best. The time spent in the different processing steps in MBC and SAV are shown in Figure 5.10. The time for bin grouping in MBC and the time for sampling in SAV are both relatively small to the initial search processing time. The time for search refinement is small in most cases, except for the case of using SAV with three fields. In this case, we needed to configure a high tolerance threshold to obtain the final result, which resulted in many false positives that needed to be removed during the refinement step.

Table 5.4: The numbers of search results for SAV.

Dataset	Isabel			Combustion			Plume		
# of fields	1	2	3	1	2	3	1	2	3
V_{final}	4184134	15709	3961	6035480	1625359	203351	10070856	599762	9928
# of true resulting voxels	4184159	15946	4113	6035497	1625359	203361	10071016	604610	10294
true positive rate applying SAV	0.999	0.985	0.963	0.999	1	0.999	0.999	0.992	0.964

5.3.2 Quality of Search Optimization

To evaluate the quality of search results after applying the proposed acceleration methods, we compare search result quantities after optimization steps in Table 5.3 and Table 5.4. As shown in the row 6 in Table 5.3, the number of voxels resulted from the search (V_{final}) after using MBC is the same as the actual search result without optimization shown in the row 7. This is due to the triangle inequality properties in the distance measurement, and therefore, there are no false negatives from the approximate result supplied by MBC. Table 5.4 shows search result comparisons when using SAV optimization. In these test cases, the sampling percentage is 20% and the p value is 1.414, like before. The majority of voxels resulted from the search using SAV are correct voxels due to the effectiveness of tuning the error tolerance via Equation 5.11. We can see that at least 96% of the correct voxels are detected by the SAV approach while reducing the computation time of searching.

We explored the relationship between the true positive hit rate during searches and the comparative performance speedup. We tested marginal feature search for the three datasets with the sampling percentages of 20%, 10%, 5% and 1%. We skipped the error tolerance tuning process to generate performance results with varying accuracies. Figure 5.8b shows that we are able to gain greater than 5 times speedup when the true positive hit rate is

Table 5.5: The run-time memory usage for the *Plume* dataset.

# of fields	# of bins	Memory usage of our method			Total raw data size
		bitmap	buffers	total	
1	105	450.299 MB	68.556 MB	518.855 MB	1930 MB
2	1018	813.079 MB	68.556 MB	881.635 MB	3860 MB
3	1321	768.547 MB	68.556 MB	837.103 MB	5790 MB

smaller than 0.96, in most cases. At a minimum, we gain at least 1.385 times speedup for all cases.

5.3.3 Run-time Memory Usage Comparison

Table 5.5 shows the run-time memory usage for our methods with different numbers of fields in the *Plume* dataset where the neighborhood size was 21^3. A traditional value query needs to store the entire dataset in memory to maximize search performance (not including the memory overhead of any additional acceleration structures). In contrast, we only need to keep the joint distribution bitmap bmp_{JD} (the bins in $\hat{\mathscr{B}}$) and two constant-sized buffers (deposit buffer and error buffer) in memory at run-time. The size of the bmp_{JD} is much smaller than the original data, and it grows slowly when the number of fields increases, as compared to the total data size.

5.4 Case Study

5.4.1 Rotating Stall Detection in Turbine Flow Simulation

The study of flow stability of a turbine engine is an important topic in aerospace engineering. Rotating stall is a type of flow instability that is subtle and difficult to detect in its early formation, but can soon become destructive to the engine if no proper measures

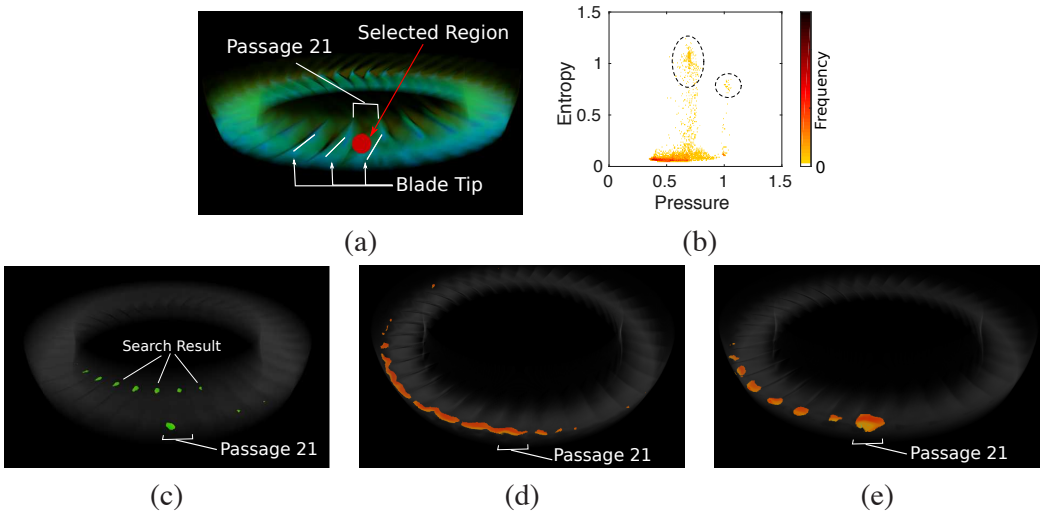


Figure 5.11: Local distribution searches for turbine flow stability analysis. (a): Selection of a stall cell region (the red sphere) close to the blade tip of passage 21. (b): Joint distribution of pressure and entropy of the selected region. (c): Search result with only the pressure distribution. (d): Search result with only the entropy distribution. (e): Search result of the joint distribution of pressure and entropy.

are taken. It is also challenging to characterize and detect the initiation of rotating stall. Several stall warning approaches have been proposed [19, 20, 31] to obtain rough ideas where and when rotating stall initiates, but its exact characteristics is still under research. In general, stall cells are identified by abnormal pressure fluctuation and high entropy values originating from regions close to blade tips [20]. Therefore, it is hypothesized that they feature unique local distributions of pressure and entropy.

In our collaboration with an domain expert in aerospace engineering, CFD simulations of airflow passing through a jet turbine engine model, NASA Compressor Stage 35, have been conducted and high-resolution pressure and entropy fields have been collected [19]. To verify the hypothesis above, we applied our distribution search method by first obtaining the target distribution from a selected region where stall cells occur, according to the

domain expert with a known stall detection method based on vortex analysis [38]. The selected region is shown by the red sphere in passage 21 in Figure 5.11a. The joint distribution of pressure and entropy in this region was created, as shown in Figure 5.11b. The circled areas indicate that those points where higher entropy occurs also contain both higher and lower pressure values. With this as the target search feature, we then performed local distribution search to locate regions with similar distributions.

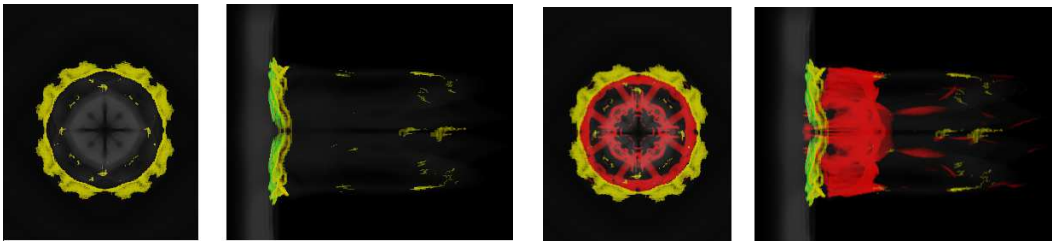
We compare the distribution search results using joint distributions and marginal distributions of both pressure and entropy values, along with a previous study [19] as the ground truth. The previous method [19] has been shown able to detect stall cells, by detecting pressure value anomalies among passages. Note that the previous method only finds anomalies of variable values from normality, which is distinctly different from our distribution search approach by locating regions with similar value compositions. Our results show that the joint distribution search on entropy and pressure can find similar regions as the previous method [19], as shown in Figure 5.11e. In addition, the joint distribution search retrieves better results than using marginal distributions on either pressure or entropy values. Figure 5.11c shows the search result of using the marginal distribution of only the pressure variable as the target distribution. As can be seen, the detected regions include the inner circle of the turbine rotor, which does not match the expected stall cell locations as earlier described. Figure 5.11d shows the result using the marginal entropy distribution. Although the detected regions better indicate stall cell locations, they also include some extra unrelated regions. Therefore, this study shows that the joint distribution of both fields can better identify stall cells than marginal distributions of a single field. Moreover, this study also shows stall cells in this dataset have similar and unique distributions of pressure and entropy values, which was not seen in the previous study. We accelerated the joint feature

search by our MBC approach. The spatial neighborhood search size of the feature was 21^3 with an error tolerance of 0.6. The search time without acceleration was 79 seconds, while with MBC it only took 30 seconds, resulting in a 2.63 times speedup.

5.4.2 Ionization Front Instability Dataset

We tested the *Ionization Front Instability* dataset from the 2008 IEEE Visualization Contest [93] to demonstrate multivariate local distribution search applied in the astrochemistry field. It simulates the formation of the first stars in the universe through exploring the instability of an ionization front. There are one velocity vector field and ten scalar fields: eight chemical species, particle density and gas temperature.

Among the eight chemical species, scientists are most interested in H_2 formation since it has a great effect in the formation of the first stars. During the H_2 formation, the H_2^+ and H^- are the key intermediate products and their peak abundances are usually accompanied with peak H_2 abundances [94]. This provides a clue that locations with high mass abundance of H_2^+ and H^- are the possible locations of H_2 formation. The scientists can quickly locate the regions that satisfy the condition of H_2 formation by means of our multivariate distribution search algorithm. In this case study, the H_2^+ and H^- variables in time step 80 are used to explore the regions of H_2 formation by applying marginal feature search approach. We selected a region that contains relatively high mass abundance of H_2^+ and H^- to construct the marginal distribution for each variable, and applied our search approach to find the matched regions. Figure 5.12a shows the search result, where the colored (yellow and green) regions are the matched regions, and the grey regions are the unmatched regions. The left image in Figure 5.12a is a view from positive X axis and the right image is a view from positive Z axis.



(a) (b)

Figure 5.12: Case study of the Ionization dataset. (a) Search result with the joint distribution of H_2^+ and H^- variables. (b) Volume rendering for the magnitude of the curl of the velocity field. Red color represents where the magnitude value is greater than 3500; grey color represents where the magnitude value is less than 3500.

Furthermore, the scientists are also interested in the relation between turbulence and H_2 formation [93]. We used the curl magnitude of the velocity field as an estimator of turbulence. In Figure 5.12b, the red regions represents where the curl magnitudes are greater than 5500 (The whole value range in this time step is 0 to 15253.4), and the yellow and green regions represent the regions with high mass abundance of H_2^+ and H^- which are also shown in Figure 5.12a. It can be observed that the red regions and the yellow green region do not overlap, which implies the lower turbulence regions (with lower curl magnitudes) are accompanied with higher mass abundance of H_2^+ and H^-. We accelerated marginal feature search by our SAV approach. The spatial neighborhood size for the feature was 10^3 with an error tolerance of 0.7. The unaccelerated search time was 87 seconds, while SAV with 10% active voxels took 29 seconds. This was with a 0.972 true positive hit rate resulting in a 3.066 times speedup.

5.5 Discussion

Pros and Cons of MBC and SAV Methods For the MBC method, it achieves a performance improvement during joint feature search and it becomes more effective when more fields are used. From Table 5.3, we can see that the number of bins of joint distribution-based features greatly increases, but the number of grouped bins do not differ much when the number of fields grows. This indicates that the number of iterations for frequency comparisons will be reduced more when more fields are used. Therefore, we conclude that MBC has greater scalability when the number of fields increases. Comparatively, SAV does not gain as much performance improvement in multi-field cases for joint feature search. This is primary due to the errors between local joint distributions constructed from full and sampled active voxels, which increases when multiple fields are taken into account. This results in increasing computational overheads to cull false positives. However, SAV method is useful in the single-field cases with large neighborhood sizes. It is especially applicable for the marginal feature search that has applied in scientific data analyses due to the ease of defining features in separate variables. Based on the properties of MBC and SAV, we suggest applying MBC to joint feature search for multi-field cases and applying SAV to marginal feature search with large neighborhood sizes.

The Invariance Properties of Feature Search Our method is primarily only translation invariant, whereas rotation and scale invariance requires additional features or additional searching on our part. Since we search for a feature at every voxel, we can guarantee that our method will find the translated features within a dataset. To cover all scales of a feature, we would need to perform a search multiple times where the feature has been explicitly scaled up or down, spatially. Our method is not scale invariant, and we need to

develop additional algorithms in the future to efficiently and implicitly search for a feature at different scales.

As for rotation invariance, our search is based on value distributions, which means that our feature definitions lack any spatial orientation to begin with, aside from block-based definitions. Therefore, to incorporate spatial orientation, we would need to update our current strategies. One method would be to search for a feature that has been rotated about all axes, sweeping it through a bounding box larger than its initial definition, and averaging its distribution in that space. Alternatively, we can create rotated versions of an initial feature, searching for each rotated version individually, but these still lack certain directional information under reflection. In the former case, the feature search size will increase to the bounding box of the rotation, slowing down the search, and the averaged distribution within the "rotation space" may not result in the desired search. In the latter case, feature search time will increase by the number of rotated versions and certain types of directional features cannot be found. In the future, we would need to allow for non-block based features, with explicit shape, and incorporate efficient rotation invariant searching.

Local Distribution Search for Multiple Fields using GPU Although GPUs can enhance the performance of many algorithms through parallelism, its hardware architecture usually limits the scope. For example with the traditional local distribution search, the small GPU memory capacity is especially difficult for computing multidimensional joint distributions. Our algorithm is currently designed for sequential computation, and it is able to achieve efficient local feature search with a small memory footprint, even for large multi-field datasets, as shown in Section 5.3.3. This is because our algorithm works on a

compressed bit vector for each bin, along with the use of region-based algorithm, as introduced in Section 5.2.6. Parallelism is possibly applied to our algorithm, with partitioned search domain. We leave the improvement of parallelism as the future work.

5.6 Summary

In this Chapter, we proposed two efficient local distribution-based feature search algorithms for multi-field datasets. The marginal feature search provides users a visual exploration of features which are described by the characteristic of each individual field. The joint feature search provides users the ability to explore features that depends on the joint characteristic of several attributes in a local region. Furthermore, we enabled efficient search process by leveraging compressed bitmap index. To perform frequency comparisons in a joint distribution, we proposed the merged-bin-comparison (MBC) algorithm to search by comparing groups of bins. Furthermore, we applied stratified sampling in the sampled-active-voxel (SAV) algorithm to reduce the workload of large feature searches by generating approximate searches. Our experimental results and case studies showed the efficiency and efficacy of our algorithms for multi-field feature searches.

Chapter 6: Information Guided Data Sampling and Optimized Data Recovering using Bitmap Indexing

6.1 Introduction

In Chapter 4 and 5, we utilize bitmap indexing to efficiently detect the local distribution-based feature. In addition to providing efficient query responses, bitmap indexing can also be applied to reduce the data size overhead with the compressed version, which can be regarded as a data representation. The benefits of representing raw data by compressed bitmaps are: (1) It can be created at simulation time, which avoids storing and transmitting full-resolution data. (2) It can efficiently respond to a value-based query, which is a common operation for data analysis and visualization. However, the effectiveness of the bitmap compression scheme based on run-length encoding is limited. Although the size of compressed bitmaps can be less than 30% in size compared with the raw data [82], it could be still too large to be stored or transferred. A viable solution to avoid storage or bandwidth limit exceeded for such large dataset is to perform data sampling before generating bitmaps. This can further reduce the bitmap size since fewer bits in a bitmap are set to 1 due to fewer samples, so that larger number of continuous 0s would appear more frequently in the bitmap, improving the effectiveness of run-length compression. Through incorporating sampling and bitmap compression schemes, a bitmap indexing-based data representation

can be created in smaller size, which is more flexible to meet the requirements of different applications.

While performing sampling before generating bitmaps, a desired property of the sampling approach is to be able to preserve the statistical characteristics of raw data within the compact subsampled data. The stratified random sampling (StRS) [24] is such an approach which draws samples from pre-cluster groups (strata). An advanced stratified random sampling (AStRS) proposed by Su et al. [81] incorporates StRS sampling and bitmap indexing to provide better characteristic-preserved subsampled data. However, there are two remaining limitations when applying existing bitmap indexing-based data representation for data exploration and visualization. First, the data complexity in each stratum is ignored when performing data sampling so that the same amount of samples are drawn from each stratum regardless of its complexity. This can lead to loss of information in the strata with high data complexity but keeping redundant information in the strata with low data complexity. Hence, there is a need to draw samples according to the information complexity in each stratum in order to present more representative samples in the subsampled dataset with compact size. Furthermore, since the characteristic-preserved sampling method is usually stored as an irregular grid data, there is a need to reconstruct the data into a regular grid volume when applying specific types of data analysis or visualization, such as volume rendering, isosurface rendering, or local statistical-based applications [58, 54, 32, 91, 92]. However, it is challenging to have a reconstructed data volume close to the original data volume in terms of the quality of samples as well as local statistical properties.

In this Chapter, we propose two approaches to tackle these two limitations for data analysis and visualization when using bitmap indexing-based data representation. First, we propose an information guided stratified sampling (IGStS) to store more representative

samples while preserving characteristics of the raw data and keep the size of the data representation small. We apply AStRS that subdivides the raw samples into several strata and draws samples from each stratum. The strata are created by partitioning the data space into multiple non-overlapping blocks and partitioning the whole data value range into multiple bins. Instead of drawing samples evenly in each block, we calculate the information entropy to examine the data complexity in the block and determine the number of samples to draw according to the entropy value. More samples are drawn from regions with higher entropy (higher data complexity) and fewer samples for regions with lower entropy (lower data complexity). Through this adaptive sampling approach, the proposed bitmap indexing-based data representation allows superior quality of data analysis and visualization while keeping the storage cost low. Second, when a regular volume with extended number of samples as needed, we propose a novel data recovery approach from irregular subsampled data that incorporates the samples' spatial information and value distribution preserved in each local region. We transform the data recovery problem to the optimal assignment problem and solve it using the Hungarian algorithm [51]. The idea is to assign a value to each spatial location based on the likelihood of each value estimated from the neighboring samples, and to keep the same value distribution as the preserved one in each local region. This ensures the value compositions and the local statistical properties in the reconstructed data are similar to those of raw data. Through solving the assignment problem, our approach produces recovered data with small errors in terms of sample values and local statistical properties and thus is able to provide qualitative data analysis and visualization.

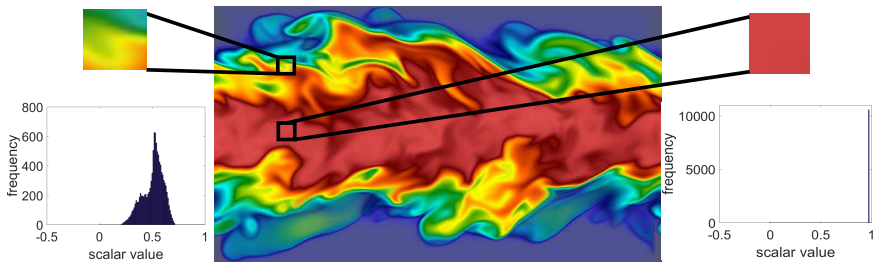


Figure 6.1: Comparisons between two data blocks with different data complexities. Drawing samples evenly in both regions may cause insufficient information stored in the data block shown in the left and redundant information stored in the right data block.

6.2 Information Guided Stratified Sampling (IGStS)

In this section, we introduce our information guided stratified sampling approach which stores representative samples that preserve the properties of the raw data and keep the sample size small simultaneously. We follow the concept of StRS used in [95] which partitions the data space into multiple non-overlapped blocks and draw samples from each block. It draws samples across the whole data volume, so the information in all the regions is ensured to be retained. However, ignoring the data complexity and drawing samples evenly over the data volume results in imbalanced information contained in the subsampled data of different region. Drawing fixed number of samples from each region can lead to information loss in regions that have high data complexity and redundant information in regions that have low data complexity. Rather than drawing the same percentage of samples from each spatial partition, we draw samples accordingly based on its data complexity, which ensures that sufficient information is stored for each region. In the mean time, the storage overhead of subsampled data is managed as compact as possible. Figure 6.1 illustrates the issue of drawing samples evenly regardless of the data complexity in each spatial partition.

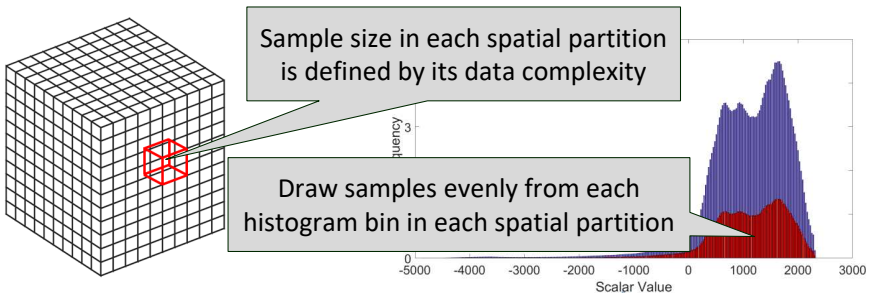


Figure 6.2: The concept of our information guided stratified sampling.

We select two different regions in the test data and display their histograms to show the data complexity. Given a fixed number of samples to be stored for each region (suppose 10 samples in this example), the region containing multiple distinct values shown in the left image cannot be well represented by only 10 samples. On the contrary, using 10 samples to represent the region with homogeneous value, like the data block shown in the right image, is unnecessary.

6.2.1 Entropy-based Stratified Sampling

We apply information theory to evaluate the data complexity for determining the number of samples that should be drawn from each local region. Shannon entropy [77] is a common approach for evaluating the information contained in the samples, which is used to determine the data complexity in our work. We divide the whole value range of the raw data into several value interval (bins) to create a sample histogram and compute the entropy value from it. Let $X = x_1, x_2, \cdots, x_n$ be the value interval for each bin, n be the number of bins, and p_{x_i} be the probability of a data value falls into x_i. The entropy of data samples

can be defined in Equation 6.1:

$$H(X) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i) \tag{6.1}$$

The minimum entropy value is 0 which occurs when all sample values fall into one value range and the maximum entropy value is $log_2 n$ which occurs if all the probabilities $p(x_i), i = 1 \cdots n$ are the same. By examining the entropy value, we are able to discriminate the data complexity between different data. A smaller entropy value indicates fewer distinct values across samples (lower data complexity) and a larger entropy value indicates more distinct values across samples (higher data complexity). In order to accurately represent the raw data using the subsampled data, it's necessary to take enough samples to cover all distinct values and retain the similar value compositions. This concept is similar to AStRS [81] which draws the same percent of the samples from each value range. In our work, we determine the sampling percentage s by entropy values, and the mapping function is defined as:

$$s = \frac{2^{H(X)}}{n} \times f \tag{6.2}$$

where f is a sampling factor to allow users to vary the sample size. The entropy value $H(X)$ can be thought of as the minimum number of bits are required on average to describe the variable X [40]. In other words, the number of events occurs in the observing entity is at least $2^{H(X)}$. For a histogram, at least $2^{H(X)}$ of bins is required to describe the variable. To define the sampling percentage, we normalize it by the total number of bins for the histogram. Once the sampling percentage is defined, we then draw samples from each value interval in order to maintain the same local histograms as that of full samples. In our work, only one sample is drawn in the homogeneous region where the entropy value is 0. Figure 6.2 shows the overall concept of our sampling approach.

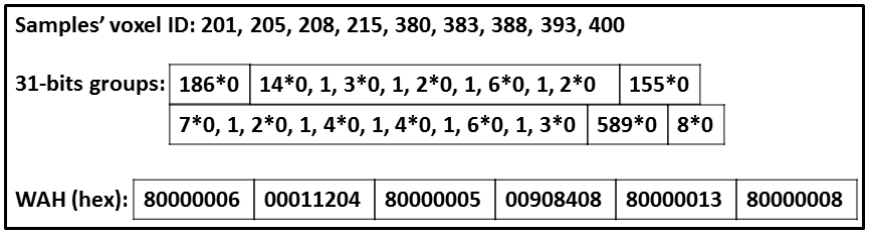


Figure 6.3: An example of creating a compressed bit vector from the samples ID using WAH32. The data size is 1000 in this example.

6.2.2 Subsampled Data Creation using Bitmap Indexing

To record samples with a small storage overhead, we create the subsampled data by utilizing compressed bitmap indexing. One bitmap is created for each spatial partition. Rather than creating a raw bitmap for the whole data, and then performing sampling on each bit vector and compressing it, we create compressed bitmap directly during sample selection. As mentioned above, while drawing samples in each partition with the sampling percentage defined above, we select a subset of samples for each value range. Samples in each group of value range can be determined when constructing the value histogram for computing the entropy value by scanning through the entire set of data points in a region. We then randomly select a number of samples from each group and store them in the order of voxel IDs. By knowing the voxel ID for each sample, the number of consecutive 0s or 1s can be easily retrieved to perform run-length encoding. Figure 6.3 shows an example of creating a compressed bit vector from the samples' voxel IDs using WAH32. Suppose the whole data size is 1000 in this example. The first line shows the voxels' ID corresponding to the samples drawn from a region. The second line shows groups with multiples of 31 bits that categorized to fill-word and literal-word, and each group is encoded accordingly in a

word (32 bits) shown in the third line. By leveraging the compressed bitmap indexing technique [99], all the bitmaps generated from all spatial partitions can be merged efficiently by applying *OR* operations between the compressed bit vectors. This makes our sampling approach suitable for large-scale scientific simulation running in the distributed system.

To further reduce the storage cost of the subsampled dataset, we apply a data reordering scheme when storing samples using bitmap indexing. As mentioned above, WAH uses run-length encoding to encode consecutive 0s or 1s and store it only by one word (32 bits), which implies it can achieve higher compression rate if more consecutive 0s or 1s appear in a bit vector. In general, the order of bits in a bit vector is the same as that of voxel IDs which follow the order of data elements stored in the disk (row-major order). However, the run length of consecutive 0s or 1s is usually short when using row-major order for creating a bit vector due to data incoherence. Instead, we rearrange the bits in block-major order when creating a bit vector, where the block size is defined by the user. Figure 6.4 shows an illustration of the difference between using row-major and block-major order for creating bitmaps. The data value retrieved from a local block with size 10^3 (shown in *A*) is uniform and the data value retrieved from one row along the *x* dimension has higher variance (shown in *B*). The bit vectors created for *A* and *B* are shown horizontally in the upper figure and lower figure, respectively. The bit set to 1 is colored black and the bit set to 0 is colored white. In each bitmap, 10 bins (*y* axis) and 1000 data points (*x* axis) are used. As shown, the bitmap for data A only has one bit vector that has bits set to 1, so we can only use 10 words to store the bitmap (1 word for 1 bit vector). On the other hand, the bitmap for data B needs more storage since bins from bin 4 to bin 8 need several words to record the discontinuities in those bit vectors. Therefore, the size of the bitmap can be reduced a lot when storing a bit vector in the block-major order.

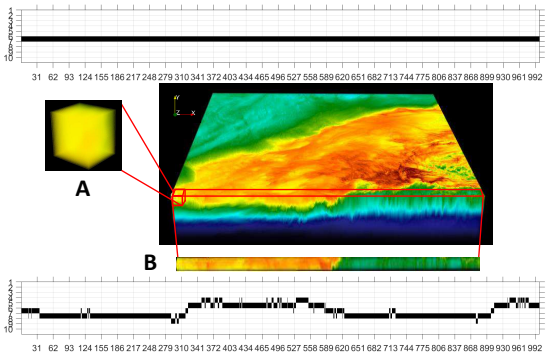


Figure 6.4: The comparisons between the bitmap generated from the data within a local block (A), and the bitmap generated from the data of one row (B).

6.2.3 Error Pre-Calculation

In our work, the sampling percentage for each spatial partition is determined by Equation 6.2. In order to allow flexible sample size to meet user requirements, we add a sampling factor f in Equation 6.2 to vary the sample size. To find out the desired sample size, we may need to compute the errors between the raw data and multiple subsampled datasets with distinct f values, which is a time-consuming process. Instead, we provide an error pre-calculation method to estimate the errors without performing sampling multiple times. For evaluating the quality of the sampled data, we consider the differences of mean, variance and entropy values between sampled data and raw data in each spatial partition as the error metrics.

After scanning through the data once, we can compute the entropy value for each spatial partition, and map it to a sampling percentage using Equation 6.2 with a given f. Thus, we know the count of samples for each bin k in each partition t ($scount_{kt}$) by the multiplication of the total count of bin k in partition t and the corresponding sampling percentage. The

mean value of the sampled data in partition t then can be determined as follows:

$$Mean_t = \frac{\sum_{k=1}^{n}(scount_{kt} \times bincenterv_k)}{\sum_{k=1}^{n} scount_{kt}} \tag{6.3}$$

n is the number of bins determined by the user, $bincenterv_k$ is the bin center value for bin k, which can be defined when knowing the data value range and the number of bins n. By varying the f value and using Equation 6.2 and 6.3, we can compute the mean value for each partition for different sample size. The variance value for each partition can be computed in a similar way and the entropy value can be computed using Equation 6.1 when knowing the probability of samples' values fall into the bin k in partition t, which is $scount_{kt} / \sum_{k=1}^{n} scount_{kt}$. After computing the errors between the raw data and the subsampled data using one of error metrics for all partitions, we can either select the maximum error or compute the average error to determine the accuracy level for each f. The user then chooses an f based on accuracy levels when performing the sampling procedure.

6.3 Recovering Data by Solving Assignment Problem

In this section, we introduce a novel data recovery approach to reconstruct the data volume for further qualitative analysis and visualization from a subsampled dataset. We transform our data recovery problem to an optimal assignment problem [60], so we first briefly introduce the assignment problem, and then introduce our data recovery approach.

6.3.1 Assignment Problem and Hungarian Algorithm

Suppose there are a number of *agents* and a number of *tasks* and we want to assign the tasks to the agents. An agent can perform any task with different cost depending on how he/she is familiar with the task. A minimum cost assignment problem is to exactly assign one agent to one task and each task is only assigned to one agent such that the total

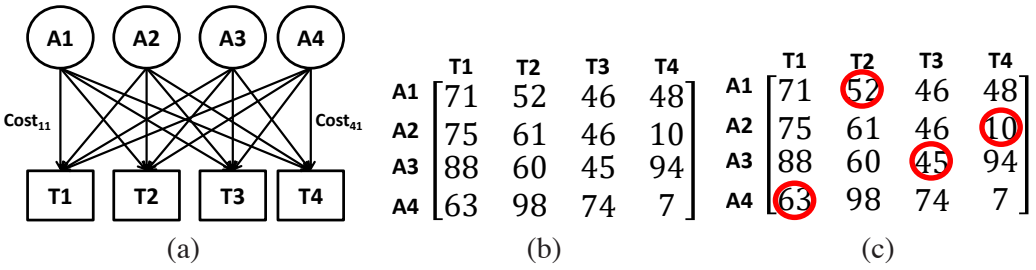


Figure 6.5: An example of an assignment problem. (a) A1, A2, A3, and A4 represent four agents and T1, T2, T3, and T4 represent four tasks. One cost value is associated with each agent and task. (b) An example of a cost matrix. (C) The assignment result for the cost matrix shown in (b)

cost of the assignment is minimized. The Hungarian algorithm [51] is the most notable approach that solves the problem in polynomial time. Figure 6.5a shows an illustration of an assignment problem which has four agents (A1 - A4) and four tasks (T1 - T4). The corresponding cost matrix is shown in Figure 6.5b, and the final assignment result is shown in Figure 6.5c. For more details of solving the Hungarian algorithm please refer to Kuhn and Yaw's article [51].

6.3.2 Data Value Assignment

To produce a reconstructed data close to the raw data, we solve the data recovery problem by applying the solution of a general assignment problem. The goal of our data recovery is to generate reconstructed data that have a similar value histogram to that of the raw data locally and globally. In this section, we introduce how to transform our data recovery problem to an assignment problem.

From our IGStS sampling approach introduced in Section 6.2, we are able to keep sample values after binning and their locations at the original full-resolution data volume

using bitmap indexing. To reconstruct the full-resolution data volume from the subsampled data, we assign one value within the whole data value range to each location where has no data value (denoted as $P_{recover}$ hereafter). To reduce the problem size, we partition the whole data value range into a number of groups (bins), and assign bin center values to $P_{recover}$ points during the reconstruction. This also matches the data structure of our sampling-based bitmap indexing that one bit vector records the points belong to the corresponding bin. Therefore, a bin center value b_i can be referred to as the *task* and a location of $P_{recover}$ can be referred to as the *agent* to form an assignment problem. To create a cost matrix, the total number of agents should be the same as the total number of tasks. In our case, the number of bins is usually smaller than the number of $P_{recover}$ points, so multiple $P_{recover}$ points are possible to be assigned by the same b_i value. Therefore, when creating a cost matrix, we duplicate columns multiple times to generate a square cost matrix. The number of duplication for each b_i is determined by the difference of the bin frequency between the the original raw data and the sampled data, $freq_{diff}(b_i)$, which is defined as $freq_{raw}(b_i) - freq_{sampled}(b_i)$. The $freq_{raw}(b_i)$ can be stored as a metadata, which could be a small storage overhead if only the non-zero bin frequencies are stored in each partition. On the other hand, we can also estimate the $freq_{raw}(b_i)$ by $freq_{sampled}(b_i)/s$, s is the sampling percentage in each region. Figure 6.6 shows an illustration of the idea above. The red bars represent $freq_{sampled}(b_i)$, and the blue bars represent $freq_{raw}(b_i)$. The yellow bar is $freq_{diff}(b_i)$ for bin i. The total number of the $P_{recover}$ points N_{tot} is defined as follows:

$$N_{tot} = \sum_{i=1}^{B} N_{assign_{b_i}} = \sum_{i=1}^{B} freq_{diff}(b_i) \tag{6.4}$$

where B is the number of bins, N_{assign_i} is the number of $P_{recover}$ points to be assigned by value b_i, and the size of the cost matrix is defined as $N_{tot} \times N_{tot}$.

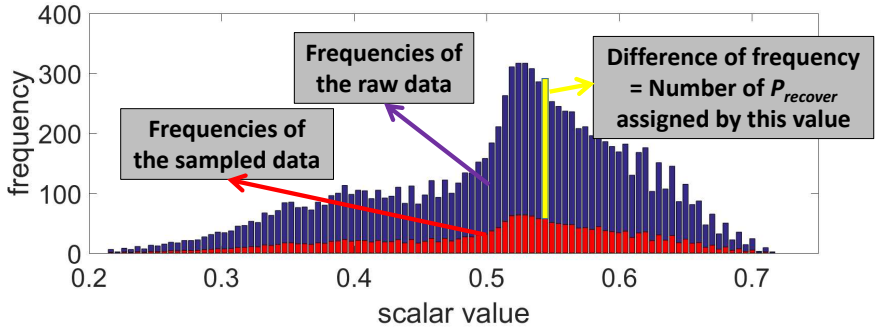


Figure 6.6: An example of determining the number of agents based on the value histograms of the subsampled data and the raw data

To determine the cost values in the cost matrix, we collect the samples around each $P_{recover}$ point and estimate the occurrence frequency of each value b_i, and assign a weighted value to the corresponding element $(P_{recovery}, b_i)$ in the cost matrix. The detail of creating the cost matrix is introduced in the next section. Our approach then assigns values to all the $P_{recover}$ points by solving the assignment problem. In this work, we transform our problem to a maximum assignment problem. To solve a maximum cost assignment problem using Hungarian algorithm, we can replace a cost value with $max(cost_i) - cost_i, i = 1, 2, \cdots, N_{tot} \cdot N_{tot}$. We use dlib library [49] to solve the assignment problem. Since the performance of solving assignment problem could be slow if N_{tot} is large, we reduce the problem size by partitioning the data space into several blocks and reconstructing the data block by block. This matches our IGStS sampling scheme and also allows our recovery process to be easily parallelized.

6.3.3 Value Occurrence Frequency Estimation for Creating Cost Matrix

To estimate the occurrence frequency of each value b_i for a $P_{recover}$ point, we take the value composition around $P_{recover}$ for reference. We collect the samples in the neighborhood region of $P_{recover}$ and count the number of samples' value belong to b_i. The reason is that each element is usually similar to its neighboring points' values especially for the scientific datasets which have smooth data continuity in general. Through the value composition, we assume a value has a higher frequency of appearance at a $P_{recover}$ point if the value appears frequently around $P_{recover}$. Furthermore, we weight the occurrence frequencies by the distance between the neighboring points and the $P_{recover}$ point. Instead of counting one for each sample, we weight the count by a Gaussian function centered at the $P_{recover}$ point. More specifically, the samples next to $P_{recover}$ have higher weight, and the samples faraway from $P_{recover}$ have lower weight when adding the weighted count into the corresponding frequency accumulator (We define the weighted count for b_i as $wcount_{b_i}$ hereafter). On the other hand, if several points around $P_{recover}$ belong to b_i, it implies that the values close to b_i should also have higher occurrence frequencies than those values distant from b_i. Therefore, we can also add $wcount_i$ into the accumulators of the bins close to b_i. For such bin b_u, the $wcount_{b_i}$ is weighted again by the distance between b_i and b_u. The weighted frequencies for a $P_{recover}$ point are then added to the corresponding elements in the cost matrix.

Each cost value $C_{j,k}^{b_i}$ (at the j_{th} row and the k_{th} column in the cost matrix) represents the weighted occurrence frequency of a bin center b_i at location j, which is defined as follows:

$$C_{j,k}^{b_i} = \sum_{m=1}^{r} e^{-\frac{1}{2}(\frac{dist(j,vm)}{\sigma})^2} \times e^{-\frac{1}{2}(\frac{diff(b_i,bu)}{\sigma})^2} \tag{6.5}$$

Here, v_m is the sample point around location j. r is the number of v_m points in the processing domain. $dist(j, v_m)$ represents the Euclidean distance between v_m and the point at j. b_u is the bin center that v_m's value belongs to. $dist(b_i, b_u)$ represents the difference between bin center b_i and b_u. σ is the standard deviation which can be defined by the user. To avoid high computational overhead of $C_{j,k}^{b_i}$, we can determine a threshold for filtering out the sample points distant from the point at j and another threshold for filtering out the sample points whose values are much different from b_i.

6.4 Comparative Studies

We perform comparative studies among different data sampling and recovery schemes to demonstrate the efficacy of the proposed approaches. In this study, we consider the storage cost, sampling rate and the quality of the reconstructed data for comparisons. Signal-to-noise ratio (SNR) is commly used to estimate the quality of the reconstructed data, and is defined as the ratio of the power of a signal to the power of the noise in the signal. Higher SNR represents better quality in the signal. Among different sampling approaches, if a sampling approach well preserves the information in the original raw data, it will result in a better quality with a higher SNR value [50]. SNR value is defined as:

$$SNR = \frac{P_{signal}}{P_{noise}} \tag{6.6}$$

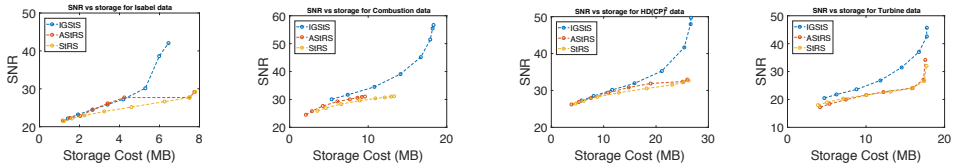
where the power of noise is calculated by the error between the original raw data and the reconstructed data. In this work, we use the logarithmic decibel scale for SNR, $SNR_{db} = 10 \times log_{10}(SNR)$.

6.4.1 Datasets

Four datasets are used in the comparison studies: The *Isabel* dataset represents Hurricane Isabel from IEEEVIS 2004 Contest, where the *Pressure* field is used. The data resolution is $500 \times 500 \times 100$ and the raw data size is 95MB. The *Combustion* dataset is a simulation of combustion phenomena provided by the Sandia National Laboratories, where the mixture fraction field is used. The data resolution is $480 \times 720 \times 120$ and the raw data size is 158MB. The $HD(CP)^2$ dataset is atmospheric simulations over Germany provided by the 2017 IEEE SciVis Contest, where the *humidity* field is used. The data resolution is $589 \times 637 \times 150$ and the raw data size is 214 MB. The *Turbine* dataset is generated from a flow simulation, TURBO [21], which will be discussed in the section of *In Situ* application study.

6.4.2 Comparative Studies for IGStS

To demonstrate the efficacy of our IGStS sampling approach using bitmap indexing, we compare our approach with StRS and AStRS which are introduced in Section 2.6. For qualitative comparisons, we generate a subsampled data from each approach and reconstruct a full-resolution data volume from the subsampled data. To make a fair comparison, we apply nearest neighbor algorithm to reconstruct data instead of our optimized data recovery since StRS is not designed for preserving the value distribution in a local region. The idea of nearest neighbor-based recovery is to search for the nearest neighboring sample around $P_{recover}$ and assign the sample's value to $P_{recover}$. Once the reconstructed data is generated, we compute the SNR value from the reconstructed data and the ground truth for all approaches, and then compare the performance of storage cost vs. SNR between them. To make a fair comparison, all the subsampled data is stored using bitmap indexing



(a) Isabel data (b) Combustion data (c) $HD(CP)^2$ data (d) Turbine data

Figure 6.7: The comparison between our data sampling approach IGStS and two other approaches, AStRS and StRS, for different datasets.

data structure with the same compression scheme. In addition, the reordering scheme introduced in Section 6.2.2 and the zlib compression for further storage reduction are used for all sampling approaches as well. In this experiment, the spatial partition size used in all approaches is $8 \times 8 \times 8$, the number of histogram bins used for generating bitmap indexing is 256. The σ used in Equation 6.5 for creating the cost matrix is set to 2.

Figure 6.7 compares three sampling approaches for the four test datasets. For our approach, we generate subsampled data with eight distinct sample sizes by adjusting f in Equation 6.2. We set f value to 0.5, 1, 2, 4, 8, 16, 32, and 64. The final sampling percentage for the whole subsampled data generated from IGStS is taken as the input sample size for StRS and AStRS, to make a fair comparison. From the figure, the SNR values are similar between the three methods when the sample size is small. However, when the sample size becomes large, the reconstructed data generated by our approach has better quality than that of AStRS or StRS in terms of SNR value. The reason is that when increasing the total sample size, it will draw more samples in those regions with high complexities, which significantly improves the representatives of the subsampled data in those regions. On the other hand, although the increase of samples may not be that significant in those regions with low complexities, the samples could still be enough to represent such regions.

For another two approaches, increasing the total number of samples will increase samples equally in each region, so the region with high data complexity can still be underrepresented, which results in low-quality data recovery in those regions. Overall, our IGStS can generate subsampled data that provide users more accurate and representative information of the raw data when exploring or visualizing the data.

6.4.3 Comparative Studies for Data Recovery

In this section, we show the performance of our data recovery method proposed in Section 6.3. In this experiment, we first generate subsampled data for each test data using the proposed IGStS approach with eight different sampling rates and then reconstruct each subsampled data to a full-resolution data volume. The sampling rates used for this test are the same as the ones we used in Section 6.4.2. The partition size for sampling procedure is $8 \times 8 \times 8$ and the number of histogram bins for generating bitmap indexes is also set to 256. In this experiment, we compare our proposed data recovery approach (denoted as RC_{cost} hereafter) with two approaches. The first one is a naive approach that assigns the average value of all sample points in the local region to each $P_{recover}$ point (denoted as RC_{mean} hereafter). The second one is to apply nearest neighbor algorithm to reconstruct data (denoted as $RC_{nearest}$ hereafter). For the RC_{cost} approach in this experiment, we store bin frequencies of the original raw data for each partition when sampling is performed, and take this metadata for evaluating the $freq_{diff}$ in Equation 6.4. The additional storage overhead for the metadata is shown in Table 6.1. As shown, the storage costs are all less than 1% of the original raw data. For all the approaches, we recover data block by block, and the block size is determined by the one used in the sampling stage. The σ in Equation 6.5 is set to 2.

Table 6.1: Additional metadata storage overhead for RC_{cost}.

	Isabel	Combustion	$HD(CP)^2$	Turbine
size (MB)	0.48	1.198	1.813	1.17
% of raw data	0.5%	0.76%	0.85%	0.08%

In this experiment, we compute an SNR value from the raw data and the reconstructed data for each recovery approach and compare their performance by SNR vs. sampling percentage. The comparisons are shown in the upper row in Figure 6.8. As shown, our approach RC_{cost} recovers data with smaller error than the approach RC_{mean} and $RC_{nearest}$ in most cases, which evidences the quality of the reconstructed data is improved compared to these approaches. Furthermore, we evaluate the quality of reconstructed data in terms of local statistical property and show the comparison results in the lower row in Figure 6.8. We compute the local histogram at each grid point for both raw data and reconstructed data, and compute the error between two histograms by the Jensen−Shannon divergence. We then create a histogram for the error values for all the grid points to evaluate the local statistical property preserved in the reconstructed data. Figure 6.8 shows the comparison results for all the dataset with the smallest sampling rate used in the previous experiments, which is 1.82% for Isabel, 4% for Combustion, 3.41% for $HD(Cp)^2$, and 5.7% for Turbine dataset. As shown, our RC_{cost} approach recovers samples with smaller errors of local histograms compared to the approach $RC_{nearest}$ and RC_{mean}, which evidences the efficacy of RC_{cost}.

6.4.4 Parameter Study

Block Size for IGStS and RC_{cost}

In this section, we compare different cases using different block sizes when applying IGStS sampling and RC_{cost} approach for the $HD(CP)^2$ and the *Combustion* dataset. In this

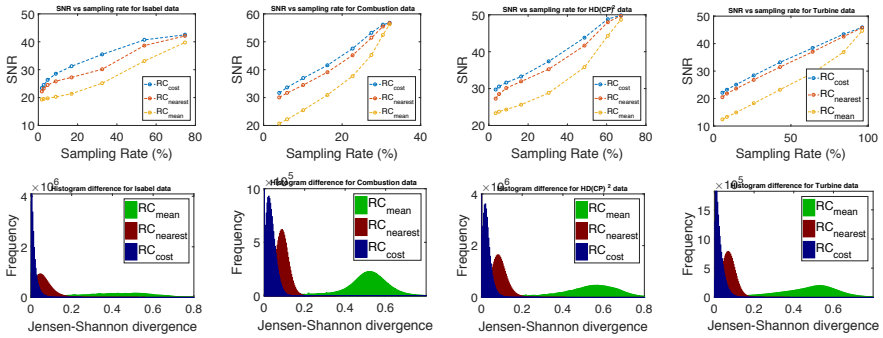
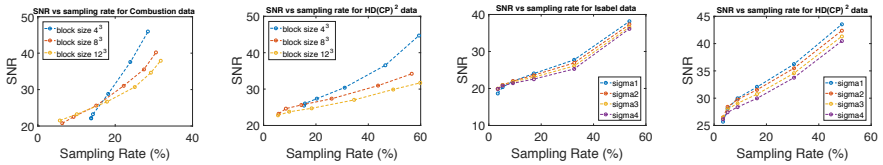


Figure 6.8: The comparison between our data recovery approach RC_{cost} and two other approaches, $RC_{nearest}$ and RC_{mean}, for different datasets.

experiment, we set the block size to 4, 8, and 12. The quality of the reconstructed data is also measured by the SNR value, sampling rate vs. SNR is compared between all results, shown in Figure 6.9a and Figure 6.9b. From the experiment, when the sampling rate is low, larger block size results in greater SNR values. This is because using a large block in the sampling procedure can provide more samples to retain more precise value distribution in each block when the sampling rate is low. On the contrary, the case with a smaller block size has a greater SNR value when the sampling rate is higher. In general, using a smaller block for storing samples can reduce the data complexity in the block. Therefore if the sample size is enough to preserve the value distribution in the block, this results in higher accuracy of estimating the bin frequencies of the raw data in the case of the bin frequencies are not stored as a metadata. Thus the RC_{cost} can recover samples closer to the raw data.

σ for RC_{cost} data recovery

In this section, we show different cases using different σ when applying the RC_{cost} data recovery approach. We set the σ value to 1, 2, 3, and 4 to gather results for $HD(CP)^2$ data and Turbine data. The σ value used in Equation 6.5 is to determine the contribution



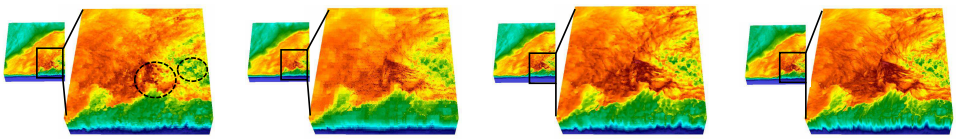
(a) (b) (c) (d)

Figure 6.9: The performance comparison of our *IGStS* and RC_{cost} approach with different parameter settings. (a) and (b) are the comparisons between three cases using different block size. (c) and (d) are the comparisons between four cases using different σ.

of a point on its neighboring points when creating a weighted histogram for estimating the value occurrence probabilities. More specifically, when setting σ to 1, only samples close to the $P_{recover}$ point can contribute large weights for the weighted histogram. On the other hand, when setting σ to 4, points far away from the $P_{recover}$ point can also contribute large weights. In this experiment, we also computed the SNR value from the reconstructed data and the raw data, and compared the SNR vs. sampling rate. Figure 6.9c and Figure 6.9d shows the experiments for different datasets. From the figures, we observe that cases with large σ have better quality results when the sampling rate is smaller. The reason is that the neighboring samples close to the $P_{recover}$ point could be too few to provide sufficient information for well estimating the occurrence probability, so we prefer to increase the weight for the samples that are far away to enhance the information. On the contrary, when the sampling rate is large, we merely consider the contributions from the neighboring samples close to the $P_{recover}$ points.

6.5 Visual Analysis

In this section, we provide visual comparisons between our approach and other sampling and recovery approaches. We apply two different sampling approaches (IGStS, and



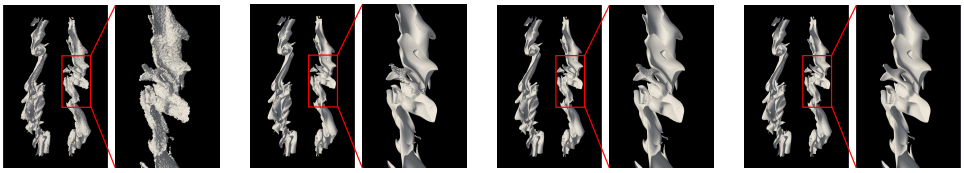
(a) (b) (c) (d)

Figure 6.10: Visual comparison of volume rendering for the $HD(CP)^2$ dataset. (a) AStS + $RC_{nearest}$ (7.0 MB)(b) IGStS + RC_{mean} (8.1 MB) (c) IGStS + RC_{cost} (7.6 MB) (d) Ground Truth (224 MB)

AStRS) to generate the subsampled data and we reconstruct each sample data by applying RC_{cost} and RC_{mean} approaches. For each dataset, the parameters used for sampling and reconstruction approaches are the same as the ones we used in the experiment in Section 6.4. We first visualize the reconstructed field by volume rendering which provides the information of the entire volume. Second, we compare the isosurface rendering results which provides a specific value information across the entire data. In this visual analysis, we use Paraview [36] to visualize all the rendering results.

6.5.1 Volume Rendering

We first apply volume rendering to visually compare the reconstructed data between our approaches and others. Figure 6.10 shows the volume rendering of the data generated from $HD(CP)^2$ project which is provided by VISContest 2017. Figure 6.10a shows the rendering result using AStRS sampling and our RC_{cost} recovery approach. Figure 6.10b shows the rendering result using our IGStS sampling and RC_{mean} recovery approach. Figure 6.10c is for the IGStS sampling and RC_{cost} approach, both of which are our proposed approaches, and Figure 6.10d shows the ground truth. For each rendering result, we zoom into the region with higher data complexity to demonstrate the performance of the proposed



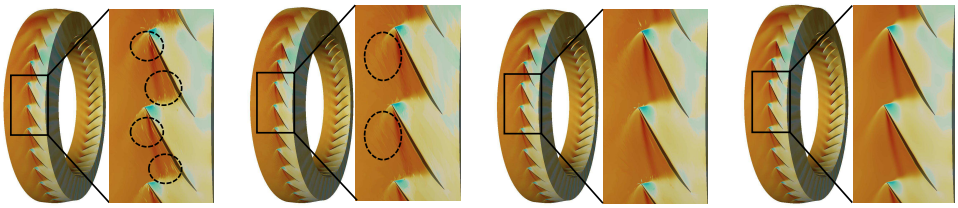
(a) (b) (c) (d)

Figure 6.11: Visual comparisons of isosurfaces 0.42 of Mixture Fraction of Combustion data. (a) AStRS + RC_{cost} (b) IGStS + RC_{mean} (c) IGStS + RC_{cost} (d) ground truth . The sampling percentage for (a) to (c) are all set to 0.16. The isosurfaces are extracted from the reconstructed fields which are generated from different sampling and recovery approaches.

approach. Figure 6.10b shows the worst rendering result compared to others. We see clear checker-box-like patterns in the rendering, and most of the detail information is missing as well. Figure 6.10a shows a better quality of rendering compared to Figure 6.10b. However, some detail information is missing in the region with higher data complexity (the regions circled by black color). Compared to Figure 6.10a, our result shown in Figure 6.10c preserves more detail information and use less storage overhead. Overall, the rendering image generated by our approaches is closest to the ground truth while the sample data size is smallest, which is only about 4% of the raw data size.

6.5.2 Isosurface Rendering

In this section, we apply isosurface rendering to visually compare the isosurface extracted from the reconstructed data between our approaches and others. Figure 6.11 shows isosurface rendering of the mixture fraction variable of the *Combustion* dataset. The mixture fraction variable represents the fraction of mass in the fuel stream, value 1 represents pure fuel and value 0 represents pure oxidizer. We set the isovalue to 0.42, which is a representation of the flame [2]. The sampling percentages used for Figure 6.11a to Figure 6.11c



(a) (b) (c) (d)

Figure 6.12: Visual comparisons of Pressure field of Turbine dataset. (a) AStRS + RC_{cost} (15.86 MB) (b) IGStS + RC_{mean} 14.94 MB (c) IGStS + RC_{cost} 14.94 MB (d) ground truth (about 290 MB)

are all 0.16. As shown, Figure 6.11a has the worst rendering result compared to others. The entire surface is jagged and unsmooth, which implies the low-quality of reconstructed data generated from the AStRS sampling. Figure 6.11b shows better isosurface rendering compared to Figure 6.11a. However, when we zoom in to the region circled by red, the structure is broken and the details disappear. Compared to 6.11b, the structure is maintained and the surface is smoother from RC_{cost} recovery, which can be seen in Figure 6.11c.

6.5.3 Surface Rendering

In this section, we visually compare the surface rendering of the Turbine dataset generated from TURBO simulations [21]. Figure 6.12 shows the rendering comparisons between different approaches, where the Pressure value is used. As shown, our approach produces a higher quality reconstruction with smaller storage overhead shown in Figure 6.12c compared to the other methods. The rendering results generated using AStRS sampling and RC_{mean} recovery approach show more artifacts highlighted by black circles in Figure 6.12a and 6.12b.

Table 6.2: In situ timings of the proposed sampling method.

Simulation	Simulation raw I/O	In situ sampling	In situ I/O
169.27 mins	3.67 mins	2.273 mins	1.45 mins

From the visualizations shown in this section, we demonstrate that our sampling approach can improve AStRS approach to better representing the raw data. We also prove that a proper data recovery approach is required to provide a qualitative post-hoc analysis and visualization from the subsampled dataset. In the next section, we will present the performance of the sampling approaches in the in situ simulations and the performance of the data recovery.

6.6 Performance Study of In Situ Sampling and Off-line Data Recovery

6.6.1 In situ Sampling Performance

To demonstrate the in situ performance, we applied our sampling method IGStS to a large scale flow simulation, TURBO [21], for data summarization. TURBO was developed at NASA and is used for studying the flow behavior in jet engine compressors. The simulation domain consists of a rotor with 36 blade passages and the spatial resolution of each passage is $151 \times 71 \times 56$. The simulation outputs 5 variables in plot3d format and they are density, momentum in x direction, momentum in y direction, momentum in z direction, and energy.

The in situ experiment was done in a cluster which contains 694 nodes with Intel Xeon x5650 CPUs (12 cores per node), and 48 GB of memory per node. The simulation was run for 1800 time steps and the in situ call was made at every 10^{th} time step. This required us

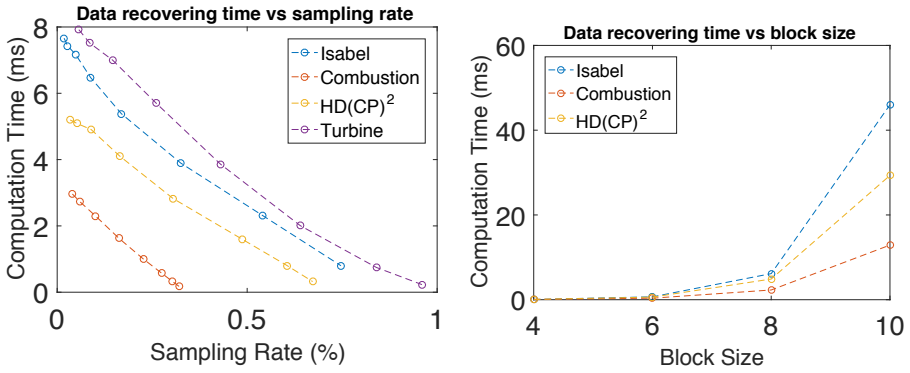


Figure 6.13: Performance comparisons for RC_{cost}. Left: computation time for each block. Right: computation time vs. block size.

to summarize data of 180 time steps. The in situ performance of our method is depicted in Table 6.2. It is observed that the in situ sampling takes significantly less time compared to the simulation run and is only a small fraction. Hence, our method does not overburden the simulation run. Furthermore, in our experiment, we have summarized all 5 variables produced by the simulation which will allow recovery of any user queried variable in the post-analysis phase. The raw data size for the rotor is 690 MB per time step, i.e., 124.2 GB for 180 time steps. In the absence of in situ summarization, we would have to store 124.2 GB data into disks for post-hoc analysis. However, using our in situ data summarization, the size of the output is reduced to only 19.79 GB which is much smaller compared to the raw data. Therefore, from the above discussion, it is evident that the proposed method is suitable for an in situ environment and can be a practical solution for efficient large scale data reduction which preserves important data properties and is able to recover data with smaller errors.

6.6.2 Off-Line Data Recovery Performance

Here we discuss the computation time of our RC_{cost} recovery approach. In this work, we apply the Hungarian algorithm to solve our data reconstruction problem. Although the time complexity of the Hungarian algorithm was improved by Munkres [60] that achieved $O(n^3)$ time complexity, still, the computation time can be slow when the number of tasks n is significantly large. To avoid this problem, we reconstruct data block by block to reduce the problem size and perform the recovery process in parallel using nVidia CUDA and OpenMP. To evaluate the performance, we ran the recovery algorithm on a machine with an Intel Core i7-4770 CPU and 16GB of system memory, and an nVidia GeForce GTX 660 GPU with 2GB of texture memory. In the test, we stored the subsampled data using the compressed bitmap indexing with the reordering scheme, and the σ used in RC_{cost} data recovery was set to 2. The performance test result of RC_{cost} is shown in Figure 6.13. As shown in the left figure, the recovery computation time for each block is less than 0.008 second for all the dataset. When setting the block size to 8^3, the total number of blocks in the test for Isabel, Combustion, $HD(CP)^2$, and Turbine dataset is 51597, 81000, 112480, and 43092 respectively. When the sampling rate is large, the number of points to be recovered is little and the size of the cost matrix is small, which results in less recovery computation time. From the figure, the recovery computation overhead for the Combustion dataset is relative small. This is because the dataset has many homogeneous blocks where only one sample is stored, and we simply assign the stored sample's value to all the points, without performing the Hungarian algorithm. Furthermore, we compare the computation time of RC_{cost} vs. block size for the Isabel and Combustion dataset. As we can see, the computation time grows rapidly when using a larger block size in the recovery procedure. Therefore, we recommend to set block size to 10^3 or smaller for data recover using RC_{cost}.

6.7 Summary

In this Chapter, we proposed two approaches to tackle the potential limitations for data analysis and visualization when using bitmap indexing-based data representation. We first introduce a new information guided stratified sampling (IGStS) technique to adaptively determine sample size for each spatial partition according to its information entropy, and then draw samples evenly from each partition's value range to preserve the characteristics of the raw data. Through this adaptive sampling approach, the proposed bitmap indexing-based data representation allows superior quality of data analysis and visualization while keeping the storage overhead low. Furthermore, we proposed an optimized data recovery algorithm that incorporates the sample's spatial information and value distribution, preserved in each spatial partition. We transform the data recovery problem to the optimal assignment problem and solve it by the Hungarian algorithm. Our experimental results showed the quantitative and visual efficacy of our two approaches. The in situ application study also demonstrated that our sampling approach is applicable in an in situ environment.

Chapter 7: Conclusion and Future Works

This dissertation has presented a framework for large-scale scalar data analysis and visualization based on data summarizations. Two types of data summarization are applied in our works. The first one is geometry data summarization which consists of a set of representative isosurfaces. The second one is compressed bitmap index which indexes data records and is stored in a bit format with compression. Several approaches using these two types of data summarization are proposed according to the scenario in the pipeline of data analysis and visualization. The summaries of our works are listed below:

- We present a novel quantitative approach to determine and evaluate the representativeness of isosurfaces in a scalar dataset by applying surface morphing and information theory. This work ensures that the meaningful information contents are kept as much as possible in the representative isosurfaces, so it is possible to perform efficient and qualitative data explorations.

- We propose an information-guided data sampling approach on a compressed bitmap to provide a qualitative sampling-based bitmap summarization. An optimized data recovery approach is also presented to improve the capability of the sampling-based bitmap when applying classical data analysis and visualization techniques for data explorations.

- We present an efficient feature detection approach using bitmap summarization to search for user-defined features described by the local histogram for enhancing the understanding of a large-scale scalar dataset. We also overcome the high demand on computation and memory footprint and propose a local histogram-based feature detection approach for the multi-field datasets.

I plan to move forward along a couple of new research directions. As discussed in Chapter 4 and Chapter 5, the proposed methods efficiently detect local-distribution based features both in single-field and multi-field dataset. Our feature search methods can detect features that are similar to the target feature regardless of rotation and translation of features. Currently, it is unable to identify features with different scales, i.e., the neighborhood size of the local distribution has to be determined priorly, and only the local distribution-based features with pre-defined size can be detected in one search process. However, an user-defined feature is occasionally presented in distinct sizes. Therefore, it can improve our understanding of the data in such case if multi-scale features can be detected. For example, in the case study of detecting rotating stall in the turbine flow simulation shown in Chapter 5, the stall cells could be presented in different sizes at different locations in the volume. A scale-invariant feature detection can assist us in determining the stall cells more precisely and even predicting the occurrences of the stall cells in other time steps.

Moreover, a novel sampling-based data summarization is proposed in Chapter 6, which can well preserve the characteristics of the raw data while keeping the size small. In the proposed approach, we partition the data space regularly into multiple blocks and draw samples from each block. However, a regular partition ignores data coherency so that high data value variance may exist in a data partition. If the data coherency is taken into account and the data space is partitioned irregularly, the data values in a data partition will be more

consistent and the data complexity can be reduced. Therefore, incorporating the irregular partition scheme, our information-guided sampling approach will preserve more precise characteristics of the raw data from the subsampled data.

Bibliography

[1] Amit Adam, Ehud Rivlin, and Ilan Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR (1)*, pages 798–805. IEEE Computer Society, 2006.

[2] Hiroshi Akiba, Kwan-Liu Ma, Jacqueline H. Chen, and Evatt R. Hawkes. Visualizing multivariate volume data from turbulent combustion simulations. *Computing in Science and Engineering*, 9(2):76–83, 2007.

[3] G. Antoshenkov. Byte-aligned bitmap compression. In *Proceedings of the Conference on Data Compression*, DCC '95, pages 476–, 1995.

[4] Chandrajit L. Bajaj, Valerio Pascucci, and Daniel R. Schikore. The contour spectrum. In *Vis '97: Proceedings of the IEEE Visualization*, VIS '97, pages 167–ff., Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[5] Stan Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *CVPR*, pages 232–237. IEEE Computer Society, 1998.

[6] Ayan Biswas, Soumya Dutta, Han-Wei Shen, and Jonathan Woodring. An information-aware framework for exploring multivariate data sets. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2683–2692, 2013.

[7] Udeepta Bordoloi and Han-Wei Shen. View selection for volume rendering. In *16th IEEE Visualization Conference, VIS 2005, Minneapolis, MN, USA, October 23-28, 2005*, pages 487–494, 2005.

[8] I. Borg and P.J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

[9] Roger Bramon, Imma Boada, Anton Bardera, Joaquim Rodriguez, Miquel Feixas, Josep Puig, and Mateu Sbert. Multimodal data fusion based on mutual information. *IEEE Trans. Vis. Comput. Graph.*, 18(9):1574–1587, 2012.

[10] Roger Bramon, Marc Ruiz, Anton Bardera, Imma Boada, Miquel Feixas, and Mateu Sbert. An information-theoretic observation channel for volume visualization. *Comput. Graph. Forum*, 32(3):411–420, 2013.

[11] Ulrik Brandes and Christian Pich. Eigensolver methods for progressive multidimensional scaling of large data. In Michael Kaufmann and Dorothea Wagner, editors, *Graph Drawing*, Lecture Notes in Computer Science, pages 42–53, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[12] David E. Breen and Ross T. Whitaker. A level-set approach for the metamorphosis of solid models. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):173–192, 2001.

[13] Stefan Bruckner and Torsten Möller. Isosurface similarity maps. *Computer Graphics Forum*, 29(3):773–782, 2010.

[14] H. Carr, B. Duffy, and B. Denby. On histograms and isosurface statistics. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1259 –1266, 2006.

[15] Hamish Carr, Jack Snoeyink, and Ulrike Axen. Computing contour trees in all dimensions. *Computational Geometry*, 24(2):75 – 94, 2003.

[16] Hamish Carr, Jack Snoeyink, and Michiel Van De Panne. Simplifying flexible isosurfaces using local geometric measures. In *Vis '04: Proceedings of the IEEE Visualization*, pages 497–504, 2004.

[17] A. Chaudhuri, T.-H. Wei, T.-Y. Lee, H.-W. Shen, and T. Peterka. Efficient range distribution query for visualizing scientific data. In *Proceedings of the 2014 IEEE Pacific Visualization Symposium (PacificVis)*, pages 201–208, 2014.

[18] Kunal N. Chaudhury and K. R. Ramakrishnan. Stability and convergence of the level set method in computer vision. *Pattern Recognition Letter*, 28(7):884–893, 2007.

[19] C. Chen, S. Dutta, X. Liu, G. Heinlein, H. Shen, and J. Chen. Visualization and analysis of rotating stall for transonic jet engine simulation. *Visualization and Computer Graphics, IEEE Transactions on*, 22(1):847–856, Jan 2016.

[20] Jen-Ping Chen, Michael D. Hathaway, and Gregory P. Herrick. Prestall Behavior of a Transonic Axial Compressor Stage via Time-Accurate Numerical Simulation. *Journal of Turbomachinery*, 130(4):041014, 2008.

[21] Jen-Ping Chen, Michael D. Hathaway, and Gregory P. Herrick. Prestall behavior of a transonic axial compressor stage via time-accurate numerical simulation. *Journal of Turbomachinery*, 130(4):041014, 2008.

[22] W. Cheung and G. Hamarneh. N-sift: N-dimensional scale invariant feature transform for matching medical images. In *Biomedical Imaging: From Nano to Macro, 2007. ISBI 2007. 4th IEEE International Symposium on*, pages 720–723, April 2007.

[23] Jerry Chou, Mark Howison, Brian Austin, Kesheng Wu, Ji Qiang, E. Wes Bethel, Arie Shoshani, Oliver Rübel, Prabhat, and Rob D. Ryne. Parallel index and query for large scale data analysis. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 30:1–30:11, New York, NY, USA, 2011. ACM.

[24] W.G. Cochran. *Sampling Techniques*. Wiley, third edition, 1977.

[25] William Gemmell Cochran. *Sampling techniques* /. Wiley series in probability and mathematical statistics. Wiley India (P.) Ltd.,, New Delhi :, 3d ed. edition, c1999.

[26] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

[27] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.

[28] Brian Duffy, Hamish Carr, and Torsten Möller. Integrating isosurface statistics and histograms. *IEEE Transactions on Visualization and Computer Graphics*, 19(2):263–277, 2013.

[29] Soumya Dutta and Han-Wei Shen. Distribution driven extraction and tracking of features for time-varying data analysis. *IEEE Trans. Vis. Comput. Graph.*, 22(1):837–846, 2016.

[30] F. Ennesser and G. Medioni. Finding waldo, or focus of attention using local color information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8), August 1995.

[31] Li Fanyu, Li Jun, Dong Xu, Sun Dakun, and Sun Xiaofeng. Stall warning approach with application to stall precursor-suppressed casing treatment. *In Proceedings of ASME Turbo Expo 2016: Turbomachinery Technical Conference and Exposition*, pages 1–9, 2016.

[32] Yi Gu and Chaoli Wang. Transgraph: Hierarchical exploration of transition relationships in time-varying volumetric data. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2015–2024, December 2011.

[33] S. Gumhold. Maximum entropy light source placement. In *ACM SIGGRAPH 2002 Conference Abstracts and Applications*, SIGGRAPH 2002, pages 215–215, New York, NY, USA, 2002. ACM.

[34] Hanqi Guo, He Xiao, and Xiaoru Yuan. Scalable multivariate volume visualization and analysis based on dimension projection and parallel coordinates. *IEEE Trans. Vis. Comput. Graph.*, 18(9):1397–1410, 2012.

[35] Martin Haidacher, Stefan Bruckner, and Meister Eduard Gr"oller. Volume analysis using multimodal surface similarity. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1969–1978, 2011.

[36] A. Henderson. Paraview guide, a parallel visualization application. 2007.

[37] P.V.C. and Hough. Machine Analysis Of Bubble Chamber Pictures. *Conf.Proc.*, C590914:554–558, 1959.

[38] D. A. Hoying, C. S. Tan, Huu Duc Vo, and E. M. Greitzer. Role of blade passage flow structures in axial compressor rotating stall inception. *Journal of Turbomachinery*, 121:735–742, 1999.

[39] Xiaolei Huang, N. Paragios, and D.N. Metaxas. Shape registration in implicit spaces using information theory and free form deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1303 –1318, 2006.

[40] Heike Jaenicke and Min Chen. An information-theoretic framework for visualization. *IEEE Transactions on Visualization & Computer Graphics*, 16:1206–1215, 2010.

[41] Heike Jänicke, Alexander Wiebel, Gerik Scheuermann, and Wolfgang Kollmann. Multifield visualization using local statistical complexity. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1384–1391, 2007.

[42] C. Ryan Johnson and Jian Huang. Distribution-driven visualization of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):734–746, 2009.

[43] Tapas Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: analysis and implementation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):881–892, Jul 2002.

[44] James R. Kent, Wayne E. Carlson, and Richard E. Parent. Shape transformation for polyhedral objects. In *SIGGRAPH '92: Proceedings of the ACM Conference on Computer Graphics and Interactive Techniques*, volume 26, pages 47–54, 1992.

[45] M.A. Khamsi and W.A. Kirk. *An Introduction to Metric Spaces and Fixed Point Theory*. Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts. Wiley, 2001.

[46] M. Khoury and R. Wenger. On the fractal dimension of isosurfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1198 –1205, 2010.

[47] Byung-Gyu Kim and Dong-Jo Park. Unsupervised video object segmentation and tracking based on new edge features. *Pattern Recognition Letters*, 25(15):1731–1742, 2004.

[48] Gordon Kindlmann and James W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *Proceedings of the 1998 IEEE Symposium on Volume Visualization*, VVS '98, pages 79–86, New York, NY, USA, 1998. ACM.

[49] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[50] U. Kohler and F. Kreuter. *Data Analysis using Stata, 2nd Edition*. StataCorp LP, 2009.

[51] H. W. Kuhn and Bryn Yaw. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–97, 1955.

[52] David H. Laidlaw, Kurt W. Fleischer, and Alan H. Barr. Partial-volume bayesian classification of material mixtures in mr volume data using voxel histograms. *IEEE Trans. Med. Imaging*, 17(1):74–86, 1998.

[53] Teng-Yok Lee, Oleg Mishchenko, Han-Wei Shen, and Roger Crawfis. View point evaluation and streamline filtering for flow visualization. In *PacificVis*, pages 83–90. IEEE Computer Society, 2011.

[54] Teng-Yok Lee and Han-Wei Shen. Efficient local statistical analysis via integral histograms with discrete wavelet transform. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2693–2702, 2013.

[55] Xiaotong Liu and Han-Wei Shen. Association analysis for visual exploration of multivariate scientific data sets. *IEEE Trans. Vis. Comput. Graph.*, 22(1):955–964, 2016.

[56] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[57] Kewei Lu and Han-Wei Shen. Multivariate volumetric data analysis and visualization through bottom-up subspace exploration. *2017 IEEE Pacific Visualization Symposium (PacificVis)*, 00:141–150, 2017.

[58] Claes Lundström, Patric Ljung, and Anders Ynnerman. Local histograms for design of transfer functions in direct volume rendering. *IEEE Trans. Vis. Comput. Graph.*, 12(6):1570–1579, 2006.

[59] Yu Ma, Xiaodong Gu, and Yuanyuan Wang. Histogram similarity measure using variable bin size distance. *Computer Vision and Image Understanding*, 114(8):981–989, 2010.

[60] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38, March 1957.

[61] Suthambhara Nagaraj, Vijay Natarajan, and Ravi S. Nanjundiah. A Gradient-Based Comparison Measure for Visual analysis of Multifield Data. *Computer Graphics Forum*, 30(3):1101–1110, 2011.

[62] Harald Obermaier and Ronald Peikert. Feature-based visualization of multifields. In Charles D. Hansen, Min Chen, Christopher R. Johnson, Arie E. Kaufman, and Hans Hagen, editors, *Scientific Visualization*, Mathematics and Visualization, pages 189–196. Springer London, 2014.

[63] Steffen Oeltze, Helmut Doleisch, Helwig Hauser, Philipp Muigg, and Bernhard Preim. Interactive visual analysis of perfusion data. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1392–1399, November 2007.

[64] Stanley Osher and James A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[65] Stanley J. Osher and Ronald P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2002.

[66] Richard E. Parent. Shape transformation by boundary representation interpolation: A recursive approach to establishing face correspondences. *The Journal of Visualization and Computer Animation*, 3(4):219–239, 1992.

[67] Greg Pass and Ramin Zabih. Comparing images using joint histograms. *Multimedia Syst.*, 7(3):234–240, May 1999.

[68] Vladimir Pekar, Rafael Wiemker, and Daniel Hempel. Fast detection of meaningful isosurfaces for volume data visualization. In *Vis '01: Proceedings of the IEEE Visualization*, pages 223–230, 2001.

[69] Simon Perreault and Patrick Hbert. Median filtering in constant time. *IEEE Transactions on Image Processing*, 16(9):2389–2394, 2007.

[70] Fatih Murat Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *CVPR (1)*, pages 829–836. IEEE Computer Society, 2005.

[71] Christof Rezk Salama, Maik Keller, and Peter Kohlmann. High-level user interfaces for transfer function design with semantics. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1021–1028, September 2006.

[72] Yann Rodriguez and Sébastien Marcel. Face authentication using adapted local binary pattern histograms. In *9th European Conference on Computer Vision (ECCV)*, 0 2006. IDIAP-RR 06-06.

[73] Oliver Rübel, E. Wes Bethel, Prabhat, and Kesheng Wu. Query-driven visualization and analysis. In E. Wes Bethel, Hank Childs, and Charles Hansen, editors, *High Performance Visualization—Enabling Extreme-Scale Scientific Insight*, Chapman & Hall, CRC Computational Science, pages 117–144. CRC Press/Francis–Taylor Group, Boca Raton, FL, USA, November 2012.

[74] Natascha Sauber, Holger Theisel, and Hans-Peter Seidel. Multifield-graphs: An approach to visualizing correlations in multifield scalar data. *IEEE Trans. Vis. Comput. Graph.*, 12(5):917–924, 2006.

[75] C.E. Scheidegger, J.M. Schreiner, B. Duffy, H. Carr, and C.T. Silva. Revisiting histograms and isosurface statistics. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1659 –1666, 2008.

[76] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th International Conference on Multimedia*, MULTIMEDIA '07, pages 357–360. ACM, 2007.

[77] C. E. Shannon. Prediction and entropy of printed english. *Bell Systems Technical Journal*, 30:50–64, 1951.

[78] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[79] Mikhail Sizintsev, Konstantinos G. Derpanis, and Andrew Hogue. Histogram-based search: A comparative study. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE Computer Society, 2008.

[80] Yu Su, Gagan Agrawal, and Jonathan Woodring. Indexing and parallel query processing support for visualizing climate datasets. In *ICPP*, pages 249–258. IEEE Computer Society, 2012.

[81] Yu Su, Gagan Agrawal, Jonathan Woodring, Kary Myers, Joanne Wendelberger, and James Ahrens. Effective and efficient data sampling using bitmap indices. *Cluster Computing*, pages 1–20, 2014.

[82] Yu Su, Yi Wang, and Gagan Agrawal. In-situ bitmaps generation and efficient data analysis based on bitmaps. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '15, pages 61–72, New York, NY, USA, 2015. ACM.

[83] Jun Tao, Jun Ma, Chaoli Wang, and Ching-Kuang Shene. A unified approach to streamline selection and viewpoint selection for 3d flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):393–406, 2013.

[84] Shivaraj Tenginakai, Jinho Lee, and Raghu Machiraju. Salient iso-surface detection with model-independent statistical signatures. In *Vis '01: Proceedings of the IEEE Visualization*, pages 231–238, 2001.

[85] David C. Thompson, Joshua A. Levine, Janine Bennett, Peer-Timo Bremer, Attila Gyulassy, Valerio Pascucci, and Philippe P. Pébay. Analysis of large-scale scalar data using hixels. In *IEEE Symposium on Large Data Analysis and Visualization, LDAV 2011, Providence, Rhode Island, USA*, pages 23–30, 2011.

[86] Pere-Pau Vázquez, Miquel Feixas, Mateu Sbert, and Wolfgang Heidrich. Automatic view selection using viewpoint entropy and its application to image-based modelling. *Computer Graphics Forum*, 22(4):689–700, Nov 2004.

[87] Ivan Viola, Miquel Feixas, Mateu Sbert, and Eduard Gröller. Importance-driven focus of attention. *IEEE Transactions on Visualization and Computer Graphics*, 12, 2006.

[88] Chaoli Wang, Hongfeng Yu, R.W. Grout, Kwan-Liu Ma, and J.H. Chen. Analyzing information transfer in time-varying multivariate data. In *Visualization Symposium (PacificVis), 2011 IEEE Pacific*, pages 99–106, March 2011.

[89] Chaoli Wang, Hongfeng Yu, and Kwan-Liu Ma. Importance-driven time-varying data visualization. *IEEE Trans. Vis. Comput. Graph.*, 14(6):1547–1554, 2008.

[90] Zhongjie Wang, Hans-Peter Seidel, and Tino Weinkauf. Multi-field pattern matching based on sparse feature sampling. *IEEE Trans. Vis. Comput. Graph.*, 22(1):807–816, 2016.

[91] Tzu-Hsuan Wei, Chun-Ming Chen, and Ayan Biswas. Efficient local histogram searching via bitmap indexing. In *Computer Graphics Forum*, volume 34, pages 81–90, 2015.

[92] Tzu-Hsuan Wei, Chun-Ming Chen, Jonathan Woodring, Huijie Zhang, and Han-Wei Shen. Efficient distribution-based feature search in multi-field datasets. In *2017 IEEE Pacific Visualization Symposium, PacificVis 2017, Seoul, South Korea, April 18-21, 2017*, pages 121–130, 2017.

[93] D. Whalen and M. L. Norman. Competition data set and description. *2008 IEEE Visualization Design Contest, http://vis.computer.org/VisWeek2008/vis/contests.html*, 2008.

[94] Daniel Whalen and Michael L. Norman. Ionization Front Instabilities in Primordial HII Regions. *Astrophys. J.*, 673:664, 2008.

[95] Jonathan Woodring, James P. Ahrens, J. Figg, Joanne Wendelberger, Salman Habib, and Katrin Heitmann. In-situ sampling of a large-scale particle simulation for interactive visualization and analysis. *Comput. Graph. Forum*, 30(3):1151–1160, 2011.

[96] Jonathan Woodring and Han-Wei Shen. Multi-variate, time varying, and comparative visualization with contextual cues. *IEEE Transactions on Visualization and Computer Graphics*, 12:909–916, September 2006.

[97] K. Wu, S. Ahern, E. W. Bethel, J. Chen, H. Childs, E. Cormier-Michel, C. G. R. Geddes, J. Gu, H. Hagen, B. Hamann, W. Koegler, J. Laurent, J. Meredith, P. Messmer, E. Otoo, V. Perevoztchikov, A. Poskanzer, Prabhat, O. Rübel, A. Shoshani, A. Sim, K. Stockinger, G. Weber, and W.-M. Zhang. Fastbit: Interactively searching massive data. *Journal of Physics Conference Series, Proceedings of SciDAC 2009*, 180:012053, June 2009.

[98] Kesheng Wu, Ekow Otoo, and Arie Shoshani. On the performance of bitmap indices for high cardinality attributes. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 24–35, 2004.

[99] Kesheng Wu, Ekow J. Otoo, and Arie Shoshani. Compressing bitmap indexes for faster search operations. In *SSDBM*, pages 99–108, 2002.

[100] Lijie Xu, Teng-Yok Lee, and Han-Wei Shen. An information-theoretic framework for flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1216–1224, November 2010.

[101] Jianlong Zhou and Masahiro Takatsuka. Automatic transfer function generation using contour tree controlled residue flow model and color harmonics. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1481–1488, 2009.