Efficient Distribution-based Feature Search in Multi-field Datasets

Tzu-Hsuan Wei* The Ohio State University
Chun-Ming Chen† The Ohio State University
Jonathan Woodring ‡ Los Alamos National Laboratory
HuiJie Zhang § Northeast Normal University
Han-Wei Shen ¶ The Ohio State University

ABSTRACT

Local distribution search is used in query-driven visualization for identifying salient features. Due to the high computational and storage costs, local distribution search in multi-field datasets is challenging. In this paper, we introduce two high performance, memory efficient algorithms for searching for local distributions that are characterized by marginal and joint features in multi-field datasets. They leverage bitmap indexing and local voting to efficiently extract regions that match a target distribution, by first approximating search results and refining to generate the final result. The first algorithm, merged-bin-comparison (MBC), reduces the computation of histogram dissimilarity measures by clustering bins. The second algorithm, sampled-active voxels (SAV), adopts stratified sampling to reduce the workload for searching local distributions with large spatial neighborhoods. The efficiency and efficacy of our algorithms are demonstrated in multiple experiments.

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 [10]. Several feature descriptors have been proposed for different search applications [4, 16, 25, 31], among which are local statistic-based approaches that define features according to the value distributions derived in local regions [9, 14, 17]. 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 [1, 5, 10, 32].

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 [19]: (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 [13] used joint features with data values and the first and second derivatives to detect boundaries in transfer function design. Pass and Zabih [20] 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 [10, 17] to avoid a high performance penalty.

In this work, we propose two independent and complementary algorithms for accelerating local distribution searches in multi-field data sets. 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 data set 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.

2 RELATED WORK

A process of understanding, visualizing, and analyzing multi-field datasets is to discover the complex relationship between different fields (variables). Several approaches have been proposed over the past decade. Sauber *et al.* [24] proposed an approach based on local correlation field to explore the relationships among multiple fields. Nagaraj *et al.* [18] proposed a gradient-based comparison measure for multi-field data. Wang *et al.* [30] explored the causal relationships by measuring information transfer among variables in time-varying multivariate datasets. These works explore the average relationships among variables, and the specific interactions among

*e-mail: wei.225@osu.edu
†e-mail: chen.1701@osu.edu
‡e-mail: woodring@lanl.gov
§e-mail: zhanghj167@nenu.edu.cn
¶e-mail: shen.94@osu.edu

2017 IEEE Pacific Visualization Symposium (PacificVis)
18-21 April, Seoul, Korea
978-1-5090-5738-2/17/$31.00 ©2017 IEEE

different values in different fields are not focused.

Other works have applied feature search techniques to discover the salient features based on scalar values among variables in multiple fields. Jänicke *et al.* [9] apply local statistical complexity to identify unique features for analyzing time-varying multifields. Johnson and Huang [10] developed boolean predicate clauses to define features based on individual local distribution for each field. Wang *et al.* [31] recently collected 3D SIFT features from each field to construct feature descriptions.

In this paper, we provide local distribution-based feature search for multiple fields. Defining a feature as a local distribution has been applied in diverse fields such as computer vision, medical image visualization, scientific data visualization and query-driven visualization. For data visualization, Laidlaw *et al.* [14] identify the material composition in a local region by constructing a local histogram within a voxel. Lundström et al. [17] proposed the partial range histogram (PRH) to identify the value range of the user-interest material by amplifying spatially coherent value ranges and then applied the PRH to perform voxel classification.

The local distribution-based feature search has been applied for single fields extensively, however, few works discuss searches for multi-field datasets. There is a performance issue, coming from requiring exhaustive local distribution comparisons to retrieve matched regions across the entire dataset. Several algorithms have been proposed to directly or indirectly solve the performance issue of local distribution search. The integral histogram has been developed [1, 15, 22] to generate local histograms in constant time. However, it requires huge storage for pre-computed integral histograms, which becomes impractical for multi-field applications. Another approach, which greatly reduces local histogram construction and search time, is to only update the histogram difference in the non-overlapped voxels from previously scanned regions [8, 21, 26]. However, the local histogram search time of these methods still grows rapidly in the multi-field case due to constructing high-dimensional local histograms at each voxel.

Wei *et al.* [32] proposed an efficient local histogram search algorithm for single-field datasets, which avoids exhaustive whole-domain search by utilizing compressed bitmap indexing [36]. The search area is iteratively reduced after comparing each histogram bin and filtering out mismatched regions. We extend the idea of utilizing bitmap indexing for local histogram search to large multi-field datasets with efficient performance and storage overhead.

3 Background

In this section, we define local distributions and describe two essential techniques our algorithm builds upon, which are bitmap indexing and the local deposit method, proposed by Wei *et al.* [32].

3.1 Local Distributions as The Feature Descriptor

A local distribution 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. A local distribution-based feature search locates voxels in a volumetric data whose local value distributions are similar to a target distribution queried by the user. In general, a distance measure, such as L_1-norm distance, is applied to compute the difference between two distributions. The search result is the set of voxels whose local distributions are within a distance threshold of the query distribution.

3.2 Bitmap Indexing

Bitmap indexing is an efficient indexing structure which has been widely applied in query-driven data visualization [23, 27, 28]. It efficiently responds to value range queries and return the indexes of the target data in a compact format. Bitmap indexing has the following two key properties that contribute to the efficiency of indexing: First, similar to a histogram, bitmap indexing divides the whole data value range into multiple small value ranges (i.e., *bins*), where each bin is associated with a bit vector encoding the occurrence of data whose values fall into this bin. Specifically in each bit vector, a bit is set to 1 if the value of the corresponding data entry falls into the value range, and 0 otherwise. In this way, the indices of a value range can be quickly found by only scanning through the associated bit vectors. Second, the word-aligned hybrid (WAH) compression [36] is applied to reduce the size of the bitmap by compressing each bit vector based on run-length encoding. The I/O latency is thus reduced in large data queries due to the compression. Meanwhile, WAH allows efficient bit-wise logical operations, such as *AND* and *OR*, among the compressed bit vectors associated with different bins.

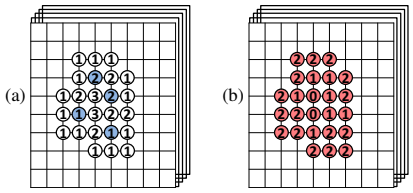


Figure 1: An example of local deposit procedure with setting neighborhood size to 3×3. (a) The blue points are the active voxels for a bin. The number label on each voxel represents the accumulated deposit counts after performing local deposit. Voxels without numbers mean that the count is zero. (b) The distance of bin_k between the target histogram with frequency 3 and the local histogram is performed only at the voxels in the search region (red circles).

3.3 Local Deposit Method

In the work proposed by Wei *et al.* [32], it is observed that users are typically interested in a partial value range during feature exploration. Therefore, feature search only needs to be applied to a subset of data value ranges. Consequently, only the frequencies of the bins within the user-defined value ranges (we define these bins as $\mathscr{B}$ hereafter) will be compared during local distribution search. To calculate the frequencies of the bins in $\mathscr{B}$ within a local neighborhood, the voxels whose values fall into those bins are located. These voxels are defined as *active voxels*:

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

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. Given a neighborhood size, only the local distributions of the voxels within the neighborhood of these active voxels will have non-zero frequencies in the bins of $\mathscr{B}$.

From this observation, an efficient feature search method is employed to quickly reduce a search region by calculating frequencies of bins in $\mathscr{B}$ with the neighborhood voxels around active voxels. Therefore, the search region can be reduced from the whole data space to just a portion of it. By leveraging compressed bitmap indexing, the active voxels for each bin are quickly retrieved and their neighboring voxels are easily located as well. Wei *et al.* proposed the *local deposit* method to calculate frequencies of bins in $\mathscr{B}$ at neighboring voxels around the active voxels, given a *deposit buffer* which is used to store the frequency counts for those bins at every voxel. Specifically, for each bin bin_k ($bin_k \in \mathscr{B}$), each active voxel that has its value in bin_k deposits one count into the corresponding frequency buffers of its neighborhood voxels. After receiving deposits from all active voxels in the neighborhood of a voxel, the accumulated counts in the deposit buffer will contain the frequencies

of bin_k around that voxel. This achieves the equivalent functionality of collecting counts of the bin_k from the neighborhood voxels, but in an inverse way (i.e., splatting rather than convolving).

Figure 1 shows an example of the local deposit method performed for one bin bin_k. The blue circles in Figure 1a represent active voxels for the bin_k. Suppose the neighborhood size is 3×3 in x and y axis. Each active voxel (blue circle) deposits (adds) one into the deposit buffers of its 3×3 neighboring voxels. In Figure 1a, the number labels are the accumulated deposit count (the frequency of the bin_k) after all active voxels have deposited values in their neighborhood voxels. The points without a number indicate that the frequencies of the bin_k are zero. After performing the local deposit, the next step is to perform feature search by comparing the frequencies with the corresponding frequencies in a target distribution for each voxel. A bin-to-bin distance measure, such as L_1-norm, is applied to measure the distance between the two distributions. As shown in Figure 1b, suppose the frequency of bin_k in the target histogram is 3, the number inside each red circle represents the difference of the frequency of bin_k between the target histogram and the local histogram at the corresponding voxel. By iteratively performing the local deposit and summing together the frequencies differences for all the bins in $\mathscr{B}$, the accumulated frequency distance of the target feature to the data set is stored in a *total distance buffer*. Each voxel in this buffer is in the final search result if the total distance is under the user-given match tolerance.

4 Local Distribution-based Feature Search in Multiple Fields

4.1 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 2. 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 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 indexing to search for distribution based features. Wei *et al.* [32] 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.

4.2 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

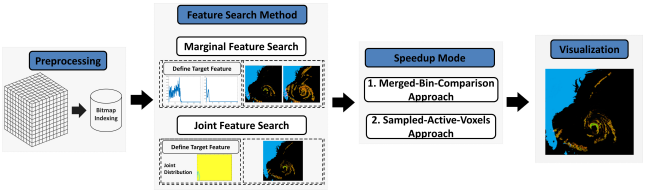


Figure 2: The flow chart of our feature search algorithm.

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 [35], 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.

4.3 Joint Feature Search for Multi-Field Datasets

To efficiently search for joint features, we also leverage bitmap indexing, 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 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:

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

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 1.

To apply the local deposit method for joint feature search, the active voxels defined in Equation 1 can be re-defined for multi-field cases as

$$V'_{active} \equiv \{v_i \mid \vec{f}(v_i) \in \text{user-interested combinations of value range from all fields}, \forall i \in \{1, \cdots, n\}\} \quad (3)$$

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.

Table 1: An example of the joint distribution based bitmap.

Dataset			Bitmap Indexing							
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	

4.4 The Merged-Bin-Comparison (MBC) Method

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.,

$$D(H_T, LH_{v_p}) = \sum_{bin_k \in \hat{\mathscr{B}}} \left| H_T(bin_k) - LH_{v_p}(bin_k) \right| \tag{4}$$

$H_T(bin_k)$ 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 4 satisfies the following inequality:

$$\sum_{bin_k} \left| H_T(bin_k) - LH_{v_p}(bin_k) \right| \geq \left| \sum_{bin_k} H_T(bin_k) - \sum_{bin_k} LH_{v_p}(bin_k) \right| \tag{5}$$

We define the left-hand side of Equation 5 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, 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, 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, $\mathscr{V}_{final}$):

$$\mathscr{V}_{app} \equiv \{v_p \mid \sum_{g=1}^{m} D_g'(H_T, LH_{v_p}) \leq \delta\} \tag{6}$$

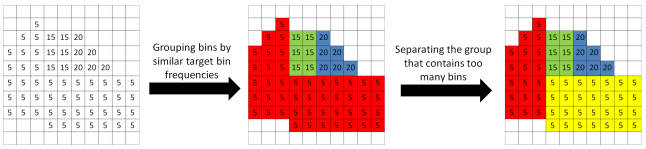


Figure 3: An example of group selection for MBC algorithm. All grids represent the 2D joint histogram of a target feature. The number in each cell is the value frequency for that bin. Bins of the same color have been grouped together. The left figure shows the joint histogram before grouping. The middle figure represents grouping bins by target frequency. The right figure represents separating the red group into two groups by bin indices.

$$\mathscr{V}_{final} \equiv \{v_p \mid \sum_{g=1}^{m} D_g(H_T, LH_{v_p}) \leq \delta\} \tag{7}$$

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 MBC 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 8 demonstrates the reduction of error between D' and D by increasing the number of groups.

$$\sum_{k=1}^{i} \left| \varphi_{bin_k} \right| \geq \left| \sum_{k=1}^{j} \varphi_{bin_k} \right| + \left| \sum_{k=j+1}^{i} \varphi_{bin_k} \right| \geq \left| \sum_{k=1}^{i} \varphi_{bin_k} \right| \tag{8}$$

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 [11] 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 3 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.

4.5 The Sampled-Active-Voxels (SAV) Method

The neighborhood size is one of the key factors for the computational cost of constructing a local histogram. In previous work [10, 16, 17], 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

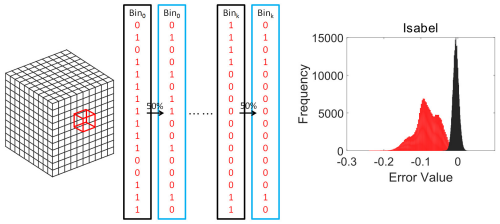


Figure 4: (a) An example of stratified sampling within a cubical region. The sampling percentage is 50% in this case. (b) The distribution of errors between approximate results and ground truth for the *Isabel* dataset. The black one is the case of stratified sampling and the red one is the case of random sampling.

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.* [29]. Compared to random sampling, searching local distributions using stratified sampled data can generate approximate results closer to the ground truth. Figure 4b 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).

4.5.1 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 4.3. To perform sampling, we select an equal percentage of samples, per value range, by creating 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 data set, by distributing samples across the value range bins, is closer to the original data set 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 4a 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.

4.5.2 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:

$$\mathscr{E}_s = \sum_{k=1}^{i} (|H_T(bin_k) - LH(bin_k)| - |H_T(bin_k) - LH_s(bin_k)|) \quad (9)$$

LH_s represents the local distribution constructed from the sampled data with the percentage s. From Equation 9, $\mathscr{E}_s$ 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,

$$\mathscr{E}_s \leq \sum_{k=1}^{i} |(|H_T(bin_k) - LH(bin_k)| - |H_T(bin_k) - LH_s(bin_k)|)| \quad (10)$$

Second, from the *reverse triangle inequality* property of absolute values [12], 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,

$$\mathscr{E}_s \leq \sum_{k=1}^{i} |LH(bin_k) - LH_s(bin_k)| \quad (11)$$

The right-hand side of Equation 11 (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

$$\delta_{tuned}(p) = \delta + \mu_s + p \times \sigma_s \quad (12)$$

4.6 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

Table 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

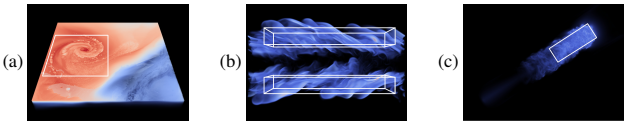


Figure 5: The selected regions for defining the target joint distribution. (a) *Isabel* (b) *Combustion* (c) *Plume* dataset.

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 [32] 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.

5 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 2, we randomly selected target local distributions to search for from the highlighted regions in Figure 5.

5.1 Performance

To compare the performance, we randomly selected voxels as search targets from the white boxed area shown in Figure 5. 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.

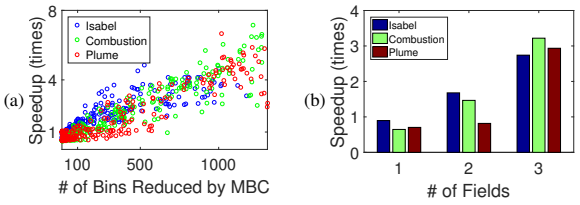


Figure 6: The performance speedup gained by our MBC approach compared to Wei *et al.*'s [32]. (a) The individual performance speedup for all test cases, where the x-axis is the number of bins reduced by MBC. (b) Grouping the test cases by data set and by the number of fields searched, showing the average performance gain.

Table 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

5.1.1 Performance Evaluation of MBC

We compare the performance of our MBC algorithm proposed in Section 4.4 to Wei *et al.*'s approach without MBC optimizations [32]. 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 data sets is shown in Figure 6. 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 6 shows the performance speedup gained from our MBC approach. Figure 6a shows all experiments (including single-field and multi-field cases) for the three data sets, 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 data set and the number of fields used to show the average performance gain in Figure 6b. 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 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 [26]. We extended their method to process 3D spatial volume data with higher dimensional joint distributions, showing the performance comparison results in Figure 7. 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 [26] in the multi-field cases. For all three data sets, 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 3.

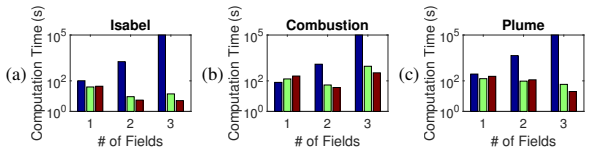


Figure 7: The performance comparisons with previous work. Blue bars: Sizintsev *et al.*. Green bars: Wei *et al.*. Red bars: our merged-bin-comparison method. Notice that the logarithmic scale is used in the y-axes. If computation took more than a day, it is shown as 10^5 seconds on this chart.

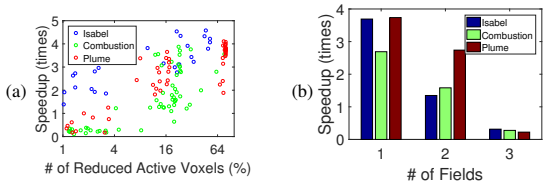


Figure 8: The performance speedup gained by our SAV method using 20% sampling, compared to Wei *et al.*'s work [32] for joint feature search. (a) All test cases where the x-axis is the percentage of total data points of the corresponding dataset. Notice the logarithmic scale for the x-axis. (b) Grouping the test cases in (a) by data set and the number of fields, showing the average performance.

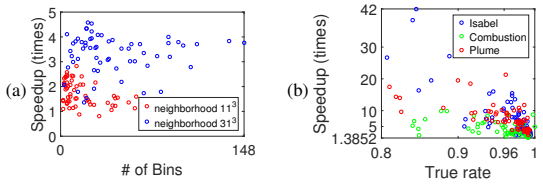


Figure 9: The performance comparison of SAV. (a) The comparison between two cases with using different neighborhood sizes. (b) The performance speedup versus accuracy of results.

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 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.

5.1.2 Performance Evaluation of SAV

We show the performance of our SAV method proposed in Section 4.5. In this experiment, we randomly selected 20 features for each data set from the white boxed areas shown in Figure 5. 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 12, was 1.414, and the associated confidence interval is 50%. The speedup performance gained by our SAV approach as compared to Wei *et al.*'s method is shown in Figure 8. Like before, we have only compared the performance gain for joint feature search in this experiment.

Figure 8a 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

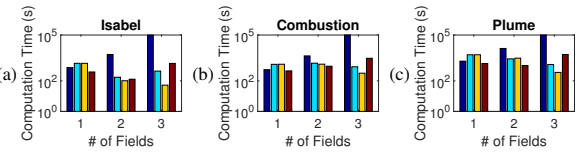


Figure 10: The computation time comparisons for joint feature searches with neighborhood size 31^3. Dark blue bar: Sizintsev *et al.*. Light blue bar: Wei *et al.*. Yellow bar: MBC. Red bar: SAV. Notice the logarithmic scale on the y-axes.

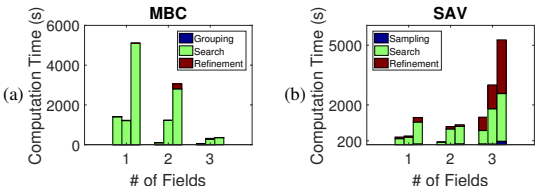


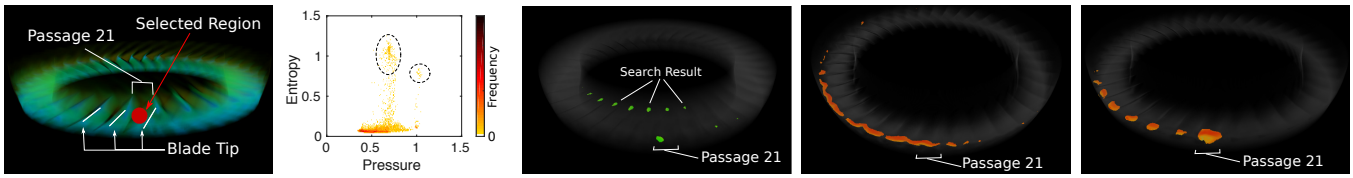
Figure 11: The computation time spent in different phases of the MBC and SAV approach, grouped by number of fields in the test. The three bars per group represent the Isabel, Combustion and Plume data sets (from left to right), respectively.

method performance increases when the number of reduced active voxels increases.

As before, we group the test cases by their corresponding data set and the number of fields to show the average performance across the groups in Figure 8b. 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 11 such that sampling is accumulated bin by bin. Therefore, the error is large when many bins are included, and it 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 9a. 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.*, Wei *et al.*, MBC, and SAV, shown in Figure 10. In all cases, our MBC method (yellow bars) or our SAV method (red bars) performs best. The time spent in the different processing steps in MBC and SAV are shown in Figure 11. 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.



(a) Dataset (b) Target distribution (c) Result of pressure only (d) Result of entropy only (e) Result of pressure & entropy

Figure 12: 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.

5.2 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 3 and Table 4. As shown in the row 6 in Table 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 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 12. 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 data sets 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 9b shows that we are able to gain greater than 5 times speedup when the true positive hit rate is smaller than 0.96, in most cases. At a minimum, we gain at least 1.385 times speedup for all cases.

5.3 Run-time Memory Usage Comparison

Table 5 shows the run-time memory usage for our methods with different numbers of fields in the *Plume* data set 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.

6 Usage Scenarios

6.1 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 are taken. It is also challenging to characterize and detect the initiation of rotating stall. Several stall warning approaches have been proposed [2, 3, 6] 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 [3]. Therefore, it is hypothesized that they feature unique local distributions of pressure and entropy.

Table 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

Table 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

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 [2]. 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 [7]. The selected region is shown by the red sphere in passage 21 in Figure 12a. The joint distribution of pressure and entropy in this region was created, as shown in Figure 12b. 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 [2] as the ground truth. The previous method [2] 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 [2], as shown in Figure 12e. In addition, the joint distribution search retrieves better results than using marginal distributions on either pressure or entropy values. Figure 12c 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 12d 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

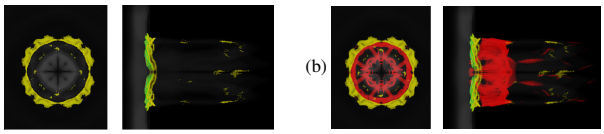


Figure 13: Case study of the Ionization dataset. (a) Search result with the joint distribution of H_2, H_2^+ and H^- variables. (b) Volume rendering for the magnitude of the curl of the velocity field. Red color represents where the magnitude value is greater than 3500; grey color represents where the magnitude value is less than 3500.

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.

6.2 Ionization Front Instability Dataset

We tested the *Ionization Front Instability* dataset from the 2008 IEEE Visualization Contest [33] 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 [34]. 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 13a 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 13a is a view from positive X axis and the right image is a view from positive Z axis.

Furthermore, the scientists are also interested in the relation between turbulence and H_2 formation [33]. We used the curl magnitude of the velocity field as an estimator of turbulence. In Figure 13b, 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 13a. 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.

7 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 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 data set. 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. 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 4.6. Parallelism is possibly applied to our algorithm, with partitioned search domain. We leave the improvement of parallelism as the future work.

8 Conclusion and Future Work

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 indexing. 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.

Currently, this work has been primarily applied to searching for features in scalar fields. We have indirectly searched for features in vector fields, by converting them to scalar fields first. In the future, we would like to adjust our methods to more directly search vector fields and tensor fields. Furthermore, our approach currently utilizes L_1-norm for distribution comparisons. We expect to see performance gains in other, but similar, bin-to-bin distance measures. This needs to be verified by future work to explore different types of distance measures for a broader range of scientific applications.

Acknowledgments

This work was supported in part by NSF grants IIS- 1250752, IIS-1065025, and US Department of Energy grants DE- SC0007444, DEDC0012495, program manager Lucy Nowell.

References

[1] 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)*, 2014.

[2] 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.

[3] J.-P. Chen, M. D. Hathaway, and G. P. Herrick. Prestall Behavior of a Transonic Axial Compressor Stage via Time-Accurate Numerical Simulation. *Journal of Turbomachinery*, 130(4):041014, 2008.

[4] 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.

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

[6] L. Fanyu, L. Jun, D. Xu, S. Dakun, and S. 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.

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

[8] T. Huang, G. Yang, and G. Tang. A fast two-dimensional median filtering algorithm. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 27(1):13–18, Feb. 1979.

[9] H. Jänicke, A. Wiebel, G. Scheuermann, and W. Kollmann. Multifield visualization using local statistical complexity. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1384–1391, 2007.

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

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

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

[13] G. Kindlmann and J. 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.

[14] D. H. Laidlaw, K. W. Fleischer, and A. 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.

[15] T.-Y. Lee and H.-W. Shen. Efficient local statistical analysis via integral histograms with discrete wavelet transform. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2693–2702, 2013.

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

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

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

[19] H. Obermaier and R. Peikert. Feature-based visualization of multifields. In C. D. Hansen, M. Chen, C. R. Johnson, A. E. Kaufman, and H. Hagen, editors, *Scientific Visualization*, Mathematics and Visualization, pages 189–196. Springer London, 2014.

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

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

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

[23] O. Rübel, E. W. Bethel, Prabhat, and K. Wu. Query-Driven Visualization and Analysis. In *High Performance Visualization—Enabling Extreme-Scale Scientific Insight*, Chapman & Hall, CRC Computational Science, pages 117–144. Nov. 2012.

[24] N. Sauber, H. Theisel, and H.-P. Seidel. Multifield-graphs: An approach to visualizing correlations in multifield scalar data. *IEEE Trans. Vis. Comput. Graph.*, 12(5):917–924, 2006.

[25] P. Scovanner, S. Ali, and M. 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.

[26] M. Sizintsev, K. G. Derpanis, and A. 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.

[27] K. Stockinger, J. Shalf, K. Wu, and E. W. Bethel. Query-driven visualization of large data sets. In *IEEE Vis '05*, pages 167–174, 2005.

[28] Y. Su, G. Agrawal, and J. Woodring. Indexing and parallel query processing support for visualizing climate datasets. In *ICPP*, pages 249–258. IEEE Computer Society, 2012.

[29] Y. Su, G. Agrawal, J. Woodring, K. Myers, J. Wendelberger, and J. Ahrens. Effective and efficient data sampling using bitmap indices. *Cluster Computing*, pages 1–20, 2014.

[30] C. Wang, H. Yu, R. Grout, K.-L. Ma, and J. Chen. Analyzing information transfer in time-varying multivariate data. In *Visualization Symposium (PacificVis), 2011 IEEE Pacific*, pages 99–106, March 2011.

[31] Z. Wang, H.-P. Seidel, and T. Weinkauf. Multi-field pattern matching based on sparse feature sampling. *IEEE Trans. Vis. Comput. Graph.*, 22(1):807–816, 2016.

[32] T.-H. Wei, C.-M. Chen, and A. Biswas. Efficient local histogram searching via bitmap indexing. In *Computer Graphics Forum*, volume 34, pages 81–90, 2015.

[33] 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.

[34] D. Whalen and M. L. Norman. Ionization Front Instabilities in Primordial HII Regions. *Astrophys. J.*, 673:664, 2008.

[35] J. Woodring and H.-W. Shen. Multi-variate, time varying, and comparative visualization with contextual cues. *IEEE Transactions on Visualization and Computer Graphics*, 12:909–916, September 2006.

[36] K. Wu, E. J. Otoo, and A. Shoshani. Compressing bitmap indexes for faster search operations. In *SSDBM*, pages 99–108, 2002.