SAND2011-2285C

Feature-Based Statistical Analysis of Combustion Simulation Data

Janine C. Bennett, Vaidyanathan Krishnamoorthy, Shusen Liu, Ray W. Grout, Evatt R. Hawkes,
Jacqueline H. Chen, Jason Shepherd, Valerio Pascucci, Peer-Timo Bremer

Abstract— We present a new framework for feature-based statistical analysis of large-scale scientific data and demonstrate its effectiveness by analyzing features from Direct Numerical Simulations (DNS). Combustion scientists use DNS to study fundamental turbulence-chemistry interactions such as extinction and auto-ignition in turbulent jet flames. Of particular interest is the scalar dissipation rate, χ, which indicates the local rate of molecular mixing, which is enhanced by turbulent flow. Turbulent strains create thin pancake-like features of locally high dissipation rate whose thickness provides a direct measure of the local mixing length-scale. Understanding the relationship between the thickness and the mean temperature within features is of principal interest to study the relationship between mechanical strains and chemical processes. This analysis is challenging due to the wide range of feature parameters that must be explored and the massive sizes of the simulation.

In our approach we precompute merge trees of the χ field which encode the set of features for all possible χ thresholds. Furthermore, we augment the merge trees with attributes, such as statistical moments of various scalar fields, *e.g.* χ, temperature, etc., as well as length scales computed via spectral analysis. The computation is performed in an efficient streaming manner in a pre-processing step and results in a collection of meta-data that is orders of magnitude smaller than the original simulation data. This meta-data is sufficient to support a fully flexible and interactive analysis of the features, allowing for arbitrary χ thresholds, providing per-feature statistics, and creating various global diagnostics such as Cummulative Density Functions (CDFs), histograms, or time-series. We combine the analysis with a rendering of the features in a linked-view browser that allows scientists to interactively explore, visualize, and analyze the equivalent of one terabyte of simulation data. While we have successfully deployed our framework to analyze statistical properties of turbulent combustion, its design and implementation are general and applicable to a wide range of scientific domains.

Index Terms—Topology, Statistics, Data analysis, Data exploration, Visualization in Physical Sciences and Engineering, Multi-variate Data.

1 Introduction

Combustion provides the vast majority of the world's energy needs and in an effort to reduce our reliance on fossil fuels, there are significant programs underway in the combustion science community to predict reliability and pollutant emissions for potential new fuel sources. To make these assessments scientists use Direct Numerical Simulations (DNS) of turbulent flames [**?**], to study effects such as flame auto-ignition [**?**] and extinction [**?**]. One of the primary drivers of these phenomena is the rate of turbulent mixing, characterized locally by the scalar dissipation rate, χ. Compressive strain in directions aligned to scalar gradients, creates thin pancake-like regions in the simulation whose thickness provides a direct measure of the local mixing length-scale. Furthermore, experimental evidence [22] suggests that thickness and mean temperature within these features are related. A more thorough understanding of this relationship would allow scientists to better characterize the effects of mechanical strain from turbulence on chemical processes and provide fundamental insights into the properties of turbulent flames.

However, this type of analysis poses several challenges: The scalar dissipation structures are typically defined using contours at locally varying isovalues [25]. Since, a wide range of values produce plausible structures, a large number of different segmentations must be explored to determine the sensitivity of the results to changes in parameters or to find stable thresholds. Furthermore, scientists are interested in subselecting based on additional criteria such as temperature variance, introducing yet more free parameters into the analysis which must be explored. Finally, many of the hypotheses are initially derived from visualizations of the temporal behavior of the flame. Thus, it is important to provide visual feedback on the impact parameter choices have on the nature of the χ structures. These challenges are multiplied by the massive size of the simulation, which in this case is roughly one terabyte.

Historically, scientists have relied on conditional statistics, applied globally, to effectively reduce the data to manageable proportions. However, even advanced indexing schemes [32, 21] are restricted to queries based on either function ranges or pre-computed properties. As will be discussed below, regions of high χ cannot be extracted through range queries. Furthermore, while pre-computing a single set of structures is feasible, the appropriate parameter choices are not known a priory and given the data size, extracting a large number of different sets for exploring the parameter space, is infeasible. Finally, traditional statistics typically provide only global averages rather than per-feature information, making simple queries such as how many features exist overall, difficult to answer.

We have developed a new integrated analysis and visualization framework to support combustion research. Our system enables a free choice of feature parameters and conditional sub-selections and interactively produces a wide range of diagnostic plots equivalent to the processing of the entire data set. Furthermore, this statistics viewer is cross-linked to a visualization of the corresponding three dimensional structures allowing picking of (sets of) features on either end. For the visualization, we use a specialized volume rendering technique optimized for sparse, dynamic, and binary segmented volumes.

Instead of extracting a single set of features we compute a multi-resolution hierarchy, capable of representing features for different parameters and at various scales. In a single pass over the original data we pre-compute a large variety of statistics for all finest resolution features. At run time the user chooses parameters resulting in a set of

- *J. C. Bennett, J. H. Chen, and J. Shepherd are with Sandia National Laboratories, Email: {jcbenne, jhchen, jfsheph}@sandia.gov.*
- *V. Krishnamoorthy, S. Liu, and V. Pascucci are with the Scientific Computing and Imaging Institute at the University of Utah, Email: {vaidy, shusen.liu, pascucci}@sci.utah.edu.*
- *R.W. Grout is with National Renewable Energy Laboratory, Email: ray.grout@nrel.gov.*
- *E.R. Hawkes is with the University of New South Wales, Email: evatt.hawkes@unsw.edu.au.*
- *P.-T. Bremer is with Lawrence Livermore National Laboratory, Email: bremer5@llnl.gov.*

Manuscript received 31 March 2011; accepted 1 August 2011; posted online 23 October 2011; mailed on 14 October 2011.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

features whose properties are aggregated on-the-fly, allowing the user to explore an entire family of feature definitions without accessing the original data. By pre-computing statistics for a base set of features, and providing the user with several multi-resolution hierarchies to explore, our system provides significantly greater flexibility in the analysis process than the typical range queries of indexing schemes. Additionally, the run-time aggregation avoids most of the costs of recomputing statistics for each set of features. As a result, our approach delivers the flexibility of extract-and-analyze techniques at an efficiency comparable to indexing or database schemes. Our system has been deployed at a National Laboratory and is actively being used to gain insight into state of the art combustion simulations. Our contributions in detail are:

- On-the-fly aggregation of feature-based spatial and temporal statistics for large scale simulations.
- An efficient encoding method for various multi-resolution hierarchies and statistics using feature-based blocked storage.
- A system for interactive creation of spatial and temporal statistical summaries including conditional empirical Cumulative Distribution Functions (CDFs), histograms, time-series, and parameter studies.
- An interactive feature browser designed using a novel volume rendering technique.
- A linked view system of statistics and features with picking and highlighting.

To demonstrate the framework we use an analysis of the relationship between length-scales and temperature of regions of high χ in turbulent combustion simulations. However, the design and implementation of our tools are general and can be applied broadly in other scientific domains where feature-based analysis is relevant.

2 Turbulent-Combustion Motivation

Combustion currently provides 85% of our nation's energy needs and will continue to be a predominant source of energy as fuel sources evolve away from traditional fossil fuels. Low emission, low-temperature engine concepts of the future operate in regimes where combustion is poorly understood. In an effort to reliably predict efficiency and pollutant emissions for new engines and fuels, computer simulations are used to study fundamental turbulence-chemistry interactions. Direct Numerical Simulations (DNS) are first principle high-fidelity computational fluid dynamics simulations in which Navier-Stokes equations are numerically solved on a computational mesh in which all of the spatial and temporal scales of the turbulence can be resolved [10]. In many practical turbulent combustion situations, turbulence strains the flame, causing molecular mixing of reactant streams. With increased mixing, chemical reactions are enhanced and overall efficiency increases up to a point, at which the loss of heat and radicals exceeds their rate of generation due to chemical reaction and the flame extinguishes resulting in increased emissions. Heat-release caused by the chemical reactions creates a complex feedback mechanism, affecting the intensity of the turbulence through density and viscosity changes across the flame.

Turbulent mixing is characterized locally by the scalar dissipation rate, χ, the rate at which scalar fluctuations decay due to diffusive processes. Compressive[1] turbulent strains create thin pancake-like regions of locally high dissipation rate. The morphology of these features, characterized according to their first three length-scales: length, width and thickness, is assumed to be correlated with length scales of turbulence. The thickness is particularly relevant as it provides a direct measure of the local mixing length-scale. Understanding the relationship between the thickness and the mean temperature within the features is of principal interest in order to study the relationship between mechanical strain and chemical processes.

There is experimental evidence [22], that the χ-layer thickness distributions are self-similar as $(T/T_0)^n$. In the the measurements of Frank and Kaiser [22], it was determined that $n \approx 0.75, T_0 \approx 400$k provided an optimal collapse of the thickness pdfs conditional on various temperatures. In this paper, we use our framework to extract the thickness pdfs and determine if the same scaling is valid for the DNS data considered here.

[1] By the term 'compressive', we mean a strain rate tensor which is compressive of scalar iso-surfaces when projected into the direction normal to the iso-surface.

3 Related work

Analysis of χ structures: Previous DNS and experimental results [16, 22, 30] have shown turbulent strain in a non-reactive jet or shear layer leads to intense mixing rate regions which are oriented by the directions of principal strain rates, characterized by relatively large dimensions in the tangent plane of the principal strain rates, and a much smaller dimension in the direction normal to the principal strain rate. Reactive flows have been studied more recently [20, 25] by directly computing and tracking χ within the simulation; however measurements were limited to thickness only and did not explore the relationship between regions of high χ and temperature.

Data warehouse technologies: At its core, our system relies on fast and efficient statistical queries for scientific data. Assuming entirely pre-computed information this problem reduces to finding and aggregating data from a large collection of records. This is a common challenge typically addressed by large Database Managment Systems (DBMSs) [11]. In such systems, each feature would be represented as one record with its corresponding statistical information collected as entries in the record. In addition to the raw data, DBMSs compute multi-dimensional search structures such as B+-trees [12] that provide efficient searches for sub-selection type queries. However, traditionally DBMSs are desgined to support constantly changing information, e.g. bank transactions, and thus their index structures have to trade query efficiency for the ability to change indices on-the-fly. On the contrary, scientific data is typically computed once and updated rarely if ever. This allows more efficient data managment relying on static indices. To distinguish such systems from DBMSs they are typically refered to as data warehouses [15, 21, 9].

One particularly successful data warehouse technology is the FastBit system [32]. Instead of search trees, FastBit relies on compressed bitmask indices [33, 34] to provide efficient subselections and can significantly out-perform other approaches [27, 28]. However, this class of data managment techniques relies almost exclusively on extracting and aggregating pre-computed information. As a result, data warehouses are well suited to access information computed for one particular set of features. In an exploratory setting, however, when the exact feature definition is unknown, warehouses are often too inflexible. While it is possible to pre-compute statistics for multiple sets of features, this is computationally expensive and the system remains limited to a small number of pre-defined feature sets. Instead, our framework uses a general feature hierarchy that allows the user to interactively change the parameters defining the features and thus explore an entire family of feature sets. Considering the inherent flexibility of common multi-resolution hierarchies, pre-computing statistics for all possible combinations of features is infeasible.

Statistics: To avoid excessive pre-processing, we exploit recent developments in parallel statistical algorithms [3] to quickly aggregate first through fourth order moments. With recent increases in data sizes there have been a number of efforts to develop robust, parallel and/or streaming statistics algorithms. Of particular interest are the centered moments and co-moments which are the building blocks of many algorithms. In [31] a single-pass algorithm for the computation of variance was developed. A more general set of pairwise update formulas for variance was introduced in [8]. The formulas for third- and fourth-order moments, which are needed to calculate skewness and kurtosis of the data set, were derived by [29]. Numerically stable, single-pass update formulas for arbitrary centered statistical moments and co-moments are presented in [3]. There also exist a number of commercial packages such as MatLab [1] and SAS [2] that support parallel statistics. By coupling, the pair-wise update formulas developed for parallel statistics computation with a general feature hierarchy, our framework provides interactive exploration of feature-based statistics.

Feature hierarchies: As discussed in Section 4, our system implements a general multi-resolution hierarchy of features [14] in which a certain number of initial high-resolution features are combined according to a scale parameter. Of particular interest in this context are a number of topology based hierarchies proposed in various settings and using a diverse algorithms. Using a variety of metrics such as fetaure volume, hyper-volume, or persistence, [7] use hierarchical contour trees [6] to define anatomical structures. Using persistence based hierarchical Morse complexes [24] show how different "resolutions" of the Morse complex encode progressively coarser segmentations of bubbles in Rayleigh-Taylor instabilities. Other examples include threshold-based hierarchies for non-premixed [25] and premixed [4] combustion simultations and core structures in porous media [17]. These techniques are particularly attractive for feature based statistics as their notion of scale corresponds to feature definitions using, for example, varying iso-surface thresholds. In general, other types of clustering methods [19] could be suited for this class of tasks.

4 Feature-Based, Statistical Analysis

The framework described in this paper is based on two linked components: Fast creation of feature-based statistics and an interactive display of the corresponding feature geometry. This section will describe the general structure of a feature-based hierarchy, the specific hierarchy used in the case study, as well as the run-time system for the creation of statistical plots.

4.1 Augmented Feature Families

One of the basic concepts of our framework is the notion of a *feature family*. Given an algorithm to define and extract features of interest corresponding to a parameter p, a feature family is a one-parameter family which for every possible parameter p stores the corresponding set of features. While any feature definition can be used to create a feature family by exhaustively pre-computing all possible features for all possible parameters, many popular algorithms natural produce nested sets of features for varying parameters. For example, clustering techniques progressively merge elements **[?]** and a threshold-based segmentation creates increasingly larger regions [5]. In such cases all features can be described either by a collection of initial features, e.g. clusters, or as a collection of differences between features at different parameters, e.g. regions above threshold a that are below threhold b.

Feature families with a nested structure can be encoded and computed in an efficient manner. In this work, we specify for each feature in the hierarchy its *life span* (in terms of the feature parameter), an arbitray number of *children*, and a single *parent*. As is common with hierarchies, the set of features at a particular parameter p is then defined as all features that are *alive* at parameter p combined with all their decendents. More formally we define:

Definition 1 (Feature)**.** *A* feature f *is defined by a unique id and minimally contains a parameter range* $[p_{min}, p_{max}]$*, a direction, a collection of id's of it children, and the id of its parent:*

$$f = (id, direction, [p_{min}, p_{max}], \{child_0, \ldots, child_n\}, parent) \in \mathbb{F}$$

The *id* is simply a unique idenifier and typically stored implicitly, *e.g.* based on the order in which features are stored in a file. The direction indicates whether a feature is *born* at $p = p_{min}$ and merges with its parent for $p \geq p_{max}$ or, inversely, is born at $p = p_{max}$ and merged for $p \leq p_{min}$. A feature family is a collection of features definedhierarchically as described above:

Definition 2 (Feature Family)**.** *A* feature family F *is a set of features*

$$F = \{f_0, \ldots, f_m\} \subset \mathbb{F}$$

Finally, in a time-dependent simulation or an ensemble of simulations we have one feature family per time or ensemble member:

Definition 3 (Clan)**.** *A* clan C *is an ordered set of feature families*

$$C = \{F_0, \ldots, F_n\} \subset \mathscr{F}$$

We store feature families in a traditional multi-resolution graph which is updated on-the-fly as the user changes parameter. At any time we maintain a set of living features that serve as the representatives for all their descendents. Furthermore, we support the encoding of multiple hierarchies associated with a feature family. In this case we simply store multiple parameter ranges and child/parent ids in each feature, one for each hierarchy. In this particular case study we use merge trees to define feature families representing either a relevance or threshold based segmentation.

Merge Tree Based Feature Families. As discussed above, scientists are interested in regions of locally high χ. As shown in [25, 5] the *merge tree* is ideally suited to hierarchically encode such regions. Given a simply connected domain $\mathbb{M}$ and a function $g : \mathbb{M} \to \mathbb{R}$ the *level set* $L(s)$ of g at isovalue s is defined as the collection of all points on $\mathbb{R}$ with function value equal to s: $L(s) = \{p \in \mathbb{M} | g(p) = s\}$. A connected component of a level set is called a *contour*. The merge tree of g represents the merging of contours as the isovalue s is swept top-to-bottom through the range of g, see Fig. 1. Each branch of the tree represents a family of contours that continuously evolve without merging as s is lowered. These contours sweep out a subset of $\mathbb{M}$ and thus the branches correspond to a segmentation of $\mathbb{M}$, see Fig. 1. To increase the resolution in parameter space we refine the merge tree by splitting its branches at regular intervals and refining the segmentation accordingly, see Fig. **??**.

Fig. 1. (a) Merge trees represent the merging of contours as f is lowered through it range. Each branch represents a portion of the domain as indicated by the colors. (b) Merging all branches above a specified function value (in color) and ignoring all below constructs a set of features (in gray).

In a simple threshold-based segmentation, the branches of the tree are features with a lifetime given by the function values of their top and bottom nodes. Given a particular threshold, each branch acts as the representative of its subtree and, by construction, each subtree represents a simply connected region of high threshold, see Fig **??**. However, when g spans multiple orders of magnitude *relevance* [25] is an alternate metric used that scales g at each node by its local maximum – the highest maximum in its corresponding subtree. The relevance lifetime of a branch is thus given by the relevance interval between its top and bottom node and ranges between 0 and 1, see Fig. **??**. To compute merge trees and their corresponding segmentation we use the streaming algorithm proposed in [5].

Feature Attributes. In addition to the information necessary to encode a feature family we augment each feature with an arbitrary number, k, of additional attributes $(att^0, \ldots, att^k)$. Our system currently supports various descriptive statistics such as minima, maxima, first through fourth order statistical moments and sums, as well as as shape descriptors such as volumes and various length-scales. The statistical information is computed on-the-fly as the merge tree is constructed using the single-pass update formulas of [3], while the shape descriptors are added in a post-processing step. For each feature f, we encode the list of domain vertices that belong to f, see Section 5. We then estimate the first three length-scales (length, width, and thickness) using a spectral technique similar to the one introduced by [26]. We parametrize each shape according to its first non-trivial eigenvector to compute its length and use the same technique recursively on iso-contours of the first eigenvector to compute the width and thickness, see Figure 2.

4.2 Interactive Exploration of Feature-Based Statistics

One of the main advantags of our system is the ability to quickly explore a wide variety of statistical information based on the given feature definitions. To achieve this our framework supports three operators that map feature families, sets of features, and statistics into new sets of features, or scalar quantities:

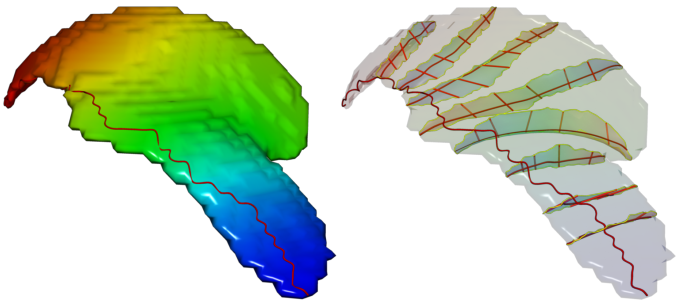

Fig. 2. The first three length-scales are estimated using a spectral technique. Each shape is parametrized according to its first non-trivial eigenvector to compute its length, and use the same technique is performed recursively on iso-contours of the first eigenvector to compute the width and thickness.

Definition 4 (Selection)**.** *A* selection $S : \mathscr{F} \times \mathbb{R} \rightarrow \mathscr{P}(\mathbb{F})$ *is an operator that given a feature family and a parameter returns a set of features as well as (a subset) of their corresponding attributes.*

Note that each feature stores attribute information regarding the portion of the domain it covers, see Fig. **??**. A selection will, for most attributes, aggregate all values in the associated subtree on-the-fly as the hierarchy is navigated. This preserves the flexibility to base different feature families on the same set of initial attributes. Nevertheless, if only one type of family is needed, aggregation of attributes can be performed once and stored to accelerate the exploration, see Section 4.3.

Definition 5 (Subselection)**.** *A* subselection $U : \mathscr{P}(\mathbb{F}) \times \{0,...,k\} \times \mathbb{R}^2 \rightarrow \mathscr{P}(\mathbb{F})$ *is an operator that given a set of features, an attribute index, and a corresponding attribute interval range, returns the subset of features whose attribute value is contained in the interval.*

The subselection operator facilitates the creation of conditional plots, which are often of significant interest to the scientists.

Definition 6 (Reduction)**.** *A* reduction $R : \mathscr{P}(\mathbb{R}) \rightarrow \mathbb{R}$ *is an operator that given a set of scalar values returns a single scalar value, for example by computing the mean.*

Using the operators described above we create three different types of plots: species distributions, parameter studies, and time-series. To simplify the discussion below, we assume that the input to each of the operators is all feature families in a clan, even though in practice we support the restriction to subsets of the data.

All plots take as input a feature clan C, a parameter p, subselections $Q = \{(att^{i_0}_{min}, att^{i_0}_{max}), \ldots, (att^{i_l}_{min}, att^{i_l}_{max})\}$, and an attribute index i. First, the parameter p is used to select an initial set of features from the clan, which are then further subselected using the subselections Q.

Species distributions plots include histograms and empirical CDFs, and track the distribution of the attribute att^i. For example, a major focus of our case study is the distribution of the mean thickness of χ structures conditioned on both the variance and mean of temperature within each feature, see Figure **??**.

A time-series, as the name suggests, shows the evolution of att^i over time, and requires an additional family-wide reduction operator, R_f, as input. For example, one might plot the maximum temperature variance of features over time. In this example R_f is the maximum of att^i which is temperature variance, see Figure **??**

Parameter studies are an extension of time-series that show how att^i changes as the parameter p is varied. For these plots a clan-wide reduction operator, R_c, is required in addition to R_f. Extending our previous example, one might plot the mean of the maximum temperature variance of features as parameter p varies. In this example R_f is maximum of the temperature variance across time steps, and R_c is the mean of these values. Note that parameter studies can be come expensive as the range and granularity of p increases, because attributes are aggregated for each p-value independently, see Figure **??**. While parameter plots are the most expensive to produce they are also often very useful. In particular, a parameter plot shows how stable or unstable a given analysis is to the parameter selection. This is crucial in any exploratory setting to guarantee that the basis of important conclusions is not an inherently unstable analysis.

We provide a convenient GUI that allows the user to specify which attributes they would like to explore, loading only those to minimize memory overhead. Subselection sliders are generated for each specified attribute automatically and, if multiple hierarchies are available, the user can toggle between these and can update parameters interactively. Optional log scaling is provided, and radio buttons are used for selection of family- and clan-wide reduction operators.

The plot viewer is linked to the feature browser described in Section 5 to provide context as statistics are explored. Only those features that have been subselected using the GUI sliders are displayed by the feature browser. Users can click on an individual feature in the feature browser to obtain details on its associated statistics. Furthermore, when the user picks regions of histograms or CDFs, only those features that are contained in the selected bins are displayed by the feature browser, see Figure 3 for an example.

4.3 File Format

We store feature families and the corresponding attributes in a modular and easily extendable file format. Typically, we save one file per feature family to easily allow the restiction to temporal subsets, for example. At the very end each file stores an XML-footer followed by the file offset to the start of the footer as the last eight bytes in the file. The XML structure encodes which components are stored for the feature family, and typically comprises a simplification sequence storing the hierarchy information in addition to a number of attributes. Any attributes stored indicate their type in addition to meta-data such as the name of the source field, how many bytes are used for each value, and whether data is stored in binary or ascii format. For the statistical moments we store not only the final value, e.g. mean, but enough information to further aggregate multiple values as needed by the parallel statistics formulas of [3]. This requires each n-th order statistical moment to store all lower-order moments to support aggregation. Most importantly the XML structure stores file offsets to each corresponding block of data, allowing for the selective loading of subsets of attributes for exploration. One immediate advantage of this file structure is that it an be easily extended without re-writing entire files. Given a new set of attributes, we read the XML footer, append the new data at the end of the old data (overwriting the old footer), update the footer, and add it to the file.

5 Interactive Feature Browser

In addition to a statistical analysis confirming or disproving a given hypothesis, our framework provides the ability to glean further insight from the data with a linked feature browser that allows the user to quickly understand how statistical trends map to the features in the domain. For each feature we provide the option of storing a list of the original domain elements that comprise the feature. This sparse data representation provides maximal flexibility, while using a minimal footprint to display the data. For example, a merge of two features amounts to a union of the two lists of domain elements. However, efficient display of features poses technical challenges:

- simulation grid sizes are prohibitively large for de-facto standard GPU accelerated volume rendering techniques [23];
- The binary segmented data is not suitable for native triliner filtering ont the GPU, rather additional effort must be taken in order to correctly perform per-fragment linear filtering; and
- the dynamic nature of the data and CPU-GPU (latency&bandwidth) bottleneck necessitates a specialized update scheme to minimize bandwidth usage.

Even though our data is sparse in nature, using a standard volume rendering method would still require a regular grid spanning the whole domain. Standard GPU accelerated raycasting techniques [23] generally use a single 3D texture to store data for a regular grid, and the size

Fig. 3. The plot viewer is linked to the feature browser providing context to the user as statistics are explored. A flexbile GUI supports subselection, toggling between hierarchies and exploration of parameters. Clicking on regions of a histogram or CDF causes only those features in the associated bins to be displayed in the feature browser.

of the simulation makes this intractable. By adopting a volume octree structure on the GPU [13], we have been able to reduce the memory usage to between 2-20% of that of a regular grid (depending on the sparsity of the input dataset). This octree structure also facilitates highly effective empty space skipping, making it possible to perform ray marching with shorter marching length thereby drastically reducing rendering time.

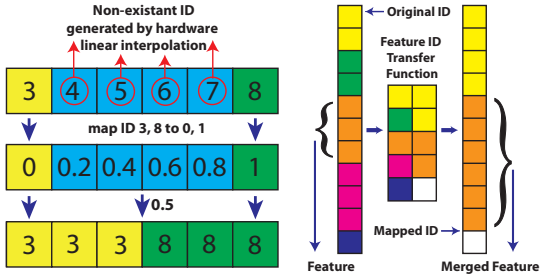

Fig. 4. The diagram on the left demonstrates interpolation between features in the GPU. Fragments are represented by rectangular blocks, and blue blocks have interpolated values. The first row shows that direct interpolation will reference incorrect feature ID. By instead mapping the neighbouring feature ID to 0-1, correct per-fragment filtering and classification is achieved. The diagram on illustrates our transfer function used to mitigate the cost of reloading feature IDs to the GPU. Rectangular blocks represent stored feature IDs (colors indicate different ID numbers, while white represents empty space). The left column shows the original feature ID volume, the right column is result of applying the transfer function showed in the middle.

Further complicating matters, each feature is represented by a set of voxels that are associated with a common feature ID and it is possible for adjacent voxels to have IDs that differ dramatically. During rendering, interpolating such IDs with a GPU's native tri-linear interpolation will give rise to very noticeable artifacts. As shown on the left of Figure 4, given two IDs (3 and 8), a direct interpolation would include multiple unapplicable feature IDs. We solve this problem by using a simple 0-1 mapping approach, as described in [18]. Adjacent voxels with different IDs are mapped to 0 (the smaller ID) and 1 (the larger ID), and then tri-linear interpolation in the 0-1 range is preformed using a shader program. We can then classify the areas with value greater than 0.5 as belonging to the larger ID and areas with value less than 0.5 as belonging to the smaller ID with per-fragment precision.

Because we are working with a multi-resolution hierarchy, the features to be displayed may merge or split and appear or disappear. From the rendering engine's perspective, the volume is constantly changing as the hierarchy is explored. To mitigate the unnecessary cost of reloading the feautre ID volume into the GPU, we introduce a special purpose feature ID transfer function generated directly from the feature hierarchy. As shown on the right side of Figure 4, the column on the left represents the original feature ID volume, while the column on the right represents the mapped feature ID. The transfer function maps a feature ID to itself, another feature ID (it's agent feature's ID) or to empty space and controls a feature's merges, splits and visibility. With the possibility of feature counts in the several hundred thousands, a regular 1D texture would be easily be rendered insufficient. We use a texture buffer object in OpenGL, capable of supporting a 1D lookup table as big as 2^{27}.

6 Results

Data. In this study the data is a temporally-evolving turbulent CO/H2 jet flame undergoing extinction and reignition at different

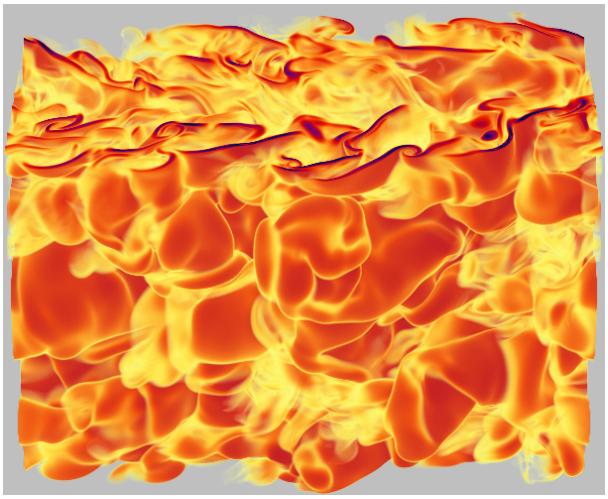

Fig. 5. The χ field from the temporally-developing CO/H2 jet flame.

Reynolds numbers [20]. The simulations were performed with up to 0.5 billion grid points and periodic boundary conditions in the mean flow (x) direction cause mixing rates to increase until approximately midway through the simulation, after which point they begin to decay. Fig. 5 shows the logarithm of χ for one of the 230 timesteps saved for postprocessing analyses.

The distribution of the χ-thicknesses shown in Figure 6 are computed for segments grouped by the mean temperature in the segment for four bins 250k wide. To ensure that the results are not confounded by within-segment temperature variation, only those segments having an approximately uniform temperature (variance of temperature ¡ 5% of the maximum) are included. The flexibility to easily and interactively add such restrictions is a key feature of our framework. The framework detects several very small features which are suspected artifacts of the segmentation algorithms; to mitigate the effects of these artifacts, which find their way into the first bin of the histogram irrespective of the bin size, the first bin in the pdfs is discarded. The remaining conditional pdfs show a trend consistent with experimental observations: the thickness distribution conditional on temperature is approximately lognormal, and shifts towards larger thickness with a broader distribution with increasing temperature. The restriction to segments of uniform temperature limits the sample size from which the distribution is drawn. Although each bin contains ¿ (how many?) samples, the exact shape of the pdf is not easy to discern.

In Figure 7, we apply the temperature scaling observed by Frank and Kaiser [22] to the conditional pdfs, by rescaling the abscissa according to:

$$t^* = t\left(\frac{T}{T_0}\right)^{-n} \tag{1}$$

with $n = 0.75$ and $T_0 = 400$ and normalizing the conditional pdfs by their peak value. As with the experimentally measured conditions, this value of n is effective to collapse the pdfs, and far less than would be expected from the value which might be expected from the dependence of the Batchelor lengthscale (a measure of the smallest scalar lengths expected in a non-reacing turbulent flow) o temperature through the kinematic viscosity (discussed in [22]). From the viscosity model used in the DNS, Batchelor scaling would suggest $n = 1.42$. In this finding, we illustrate how our framework can be used to interrogate DNS data to corroborate experimental observations in a quantitative way in addition to delivering the link to the qualitative features necessary to

develop mechanistic understanding of the phenomena.

- provide details on original data sizes + reduced file formats
- proivde timing details where appropriate (i.e. preprocessing step)

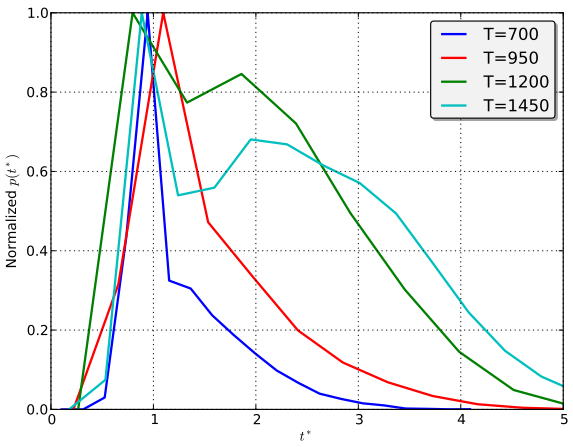

Fig. 6. The distribution of the χ-thicknesses are computed for segments grouped by the maximum temperature int the mean temperature in the segment.

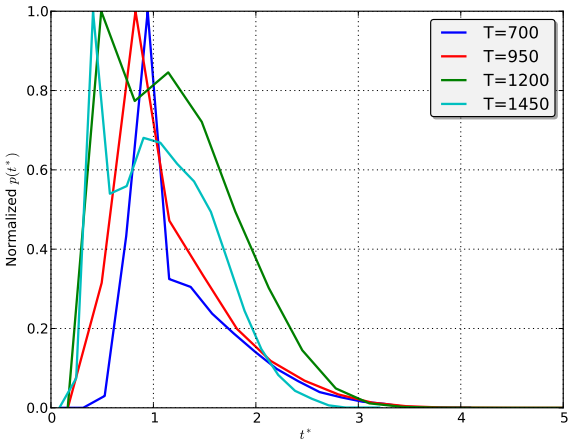

Fig. 7.

7 Conclusion and Future Work

We have presented a novel framework that supports feature-based statistical analysis of large scientific data and have successfully deployed it to glean insight regarding fundamental turbulence-chemistry interactions from combustion simulation data. Our framework comprises three main components:

- An efficient encoding scheme and data format that support multiple multi-resolution hierarchies augmented with statistics of interest.
- A statistics engine that supports on-the-fly generation and display of global trends in the data, including conditional CDFs, histograms, time series and parameter studies.
- A feature browser, built using a new volume rendering technique, that links to the statistics engine, providing context as conditional statistics are explored.

Overall, our framework delivers the flexibility of previous extract-and-analyze methods at an efficiency comparable to indexing or database schemes, and provides a natural and intuitive work-flow for the exploration of global trends in feature-based statistics.

In the future we will extend our encoding scheme to support feature hierarchies generated by other topological analysis techniques, including Reeb graphs and the Morse-Smale complex. We will support additional statistical analysis capabilities including contingency statistics, multi-correlative statistics, and principal component analysis. Our framework has been designed in a componentized manner and we plan to add additional displays, providing further context for the user. These include a display of the topological hierarchy as well as feature tracking graphs. Finally, we plan to extend our encoding scheme to support distributed data structures and parallel files so that our framework can be deployed in situ, rather than as a post process.

Acknowledgments

The authors wish to thank David Thompson, Philippe Pébay and Ajith Mascarenhas for useful discsussions.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

This work was performed under the auspices of the US Department of Energy (DOE) by the Lawrence Livermore National Laboratory under Contract Nos. DE-AC52- 07NA27344, LLNL-JRNL-412904L.

References

[1] http://www.mathworks.com/.

[2] http://www.sas.com/.

[3] J. Bennett, P. Pébay, D. Roe, and D. Thompson. Numerically stable, single-pass, parallel statistics algorithms. In *Proc. 2009 IEEE International Conference on Cluster Computing*, New Orleans, LA, Aug. 2009.

[4] P.-T. Bremer, G. Weber, V. Pascucci, M. Day, and J. Bell. Analyzing and tracking burning structures in lean premixed hydrogen flames. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):248–260, 2010.

[5] P.-T. Bremer, G. H. Weber, V. Pascucci, M. Day, and J. B. Bell. Analyzing and tracking burning structures in lean premixed hydrogen flames. *Visualization and Computer Graphics, IEEE Transactions on*, 16(2):248–260, Mar.-Apr. 2010.

[6] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Comput. Geom. Theory Appl.*, 24(3):75–94, 2003.

[7] H. Carr, J. Snoeyink, and M. van de Panne. Simplifying flexible isosurfaces using local geometric measures. In *IEEE Visualization '04*, pages 497–504. IEEE Computer Society, 2004.

[8] T. F. Chan, G. H. Golub, and R. J. LeVeque. Updating formulae and a pairwise algorithm for computing sample variances. Technical Report STAN-CS-79-773, Stanford University, Department of Computer Science, 1979.

[9] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, 26(1):65–74, 1997.

[10] J. H. Chen, A. Choudhary, B. de Supinski, M. DeVries, E. R. Hawkes, S. Klasky, W. K. Liao, K. L. Ma, J. Mellor-Crummey, N. Podhorszki, R. Sankaran, S. Shende, and C. S. Yoo. Terascale direct numerical simulations of turbulent combustion using S3D. *Computational Science & Discovery*, 2(1):015001, 2009.

[11] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 19070.

[12] D. Comer. The ubiquitous B-tree. *Computing Surveys*, 11(2):121–137, 1979.

[13] C. Crassin, F. Neyret, S. Lefebvre, and E. Eisemann. Gigavoxels: ray-guided streaming for efficient and detailed voxel rendering. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, I3D '09, pages 15–22, New York, NY, USA, 2009. ACM.

[14] E. Danovaro, L. D. Floriani, P. Magillo, E. Puppo, and D. Sobrero. Level-of-detail for data analysis and exploration: A historical overview and some new perspectives. *Computers and Graphics*, 30(3):334–344, 2006.

[15] B. A. Devlin and P. T. Murphy. An architecture for a buisiness and information system. *IBM Systems Journal*, 27(1):60–80, 1988.

[16] R. W. Grout, J. Chen, E. R. Hawkes, A. Mascarenhas, P.-T. Bremer, and V. Pascucci. Thirty-second international symposium on combustion, 2008.

[17] A. Gyulassy, M. Duchaineau, V. Natarajan, V. Pascucci, E. Bringa, A. Higginbotham, and B. Hamann. Topologically clean distance fields. *IEEE Transactions on Computer Graphics and Visualization (TVCG)*, 13(6):1432–1439, 2007.

[18] M. Hadwiger, C. Berger, and H. Hauser. High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 40–, Washington, DC, USA, 2003. IEEE Computer Society.

[19] J. Hartigan. *Clustering Algorithms*. Wiley, 1975.

[20] E. R. Hawkes, R. Sankaran, J. C. Sutherland, and J. H. Chen. Scalar mixing in direct numerical simulations of temporally evolving plane jet flames with skeletal co/h2 kinetics. *Proceedings of the Combustion Institute*, 31(1):1633 – 1640, 2007.

[21] W. H. Immon. The data warehouse and data mining. *Communications of the ACM*, 39(11):49–50, 1996.

[22] S. A. Kaiser and J. H. Frank. Imaging of dissipative structures in the near field of a turbulent non-premixed jet flame. *Proceedings of the Combustion Institute*, 31(1):1515–1523, 2007.

[23] J. Kruger and R. Westermann. Acceleration techniques for gpu-based volume rendering. *Visualization Conference, IEEE*, 0:38, 2003.

[24] D. Laney, P.-T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci. Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Trans. Visualization and Computer Graphics (TVCG) / Proc.of IEEE Visualization*, 12(5):1052–1060, 2006.

[25] A. Mascarenhas, R. W. Grout, P.-T. Bremer, E. R. Hawkes, V. Pascucci, and J. H. Chen. *Topological feature extraction for comparison of terascale combustion simulation data*, pages 229–240. Mathematics and Visualization. Springer, 2010. to appear.

[26] M. Reuter, F.-E. Wolter, M. Shenton, and M. Niethammer. Laplace-beltrami eigenvalues and topological features of eigenfunctions for statistical shape analysis. *Computer-Aided Design*, 41(10):739–755, 2009.

[27] J. Schlosser and M. Rarey. Beyond the virtual screening paradigm: Structure-based searching for new lead compounds. *J. Chemical Information and Modeling*, 49(4):800–809, 2009.

[28] T. L. L. Siqueira, R. R. Ciferri, V. C. Times, and C. D. de Aguiar Ciferri. A spatial bitmap-based index for geographical data warehouses. In *Proc. SAC*, pages 1336–1342, 2009.

[29] T. B. Terriberry. Computing higher-order moments online, 2008. http://people.xiph.org/ tterribe/notes/homs.html.

[30] P. Vaishnavi, A. Kronenburg, and C. Pantano. On the spatial resolution for scalar dissipation measurement in turbulent jet flames. *J. Fluid Mech.*, 596:103–132, 2008.

[31] B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

[32] K. Wu, S. Ahern, E. W. Bethel, J. Chen, H. Childs, E. Cormier-Michel, C. Geddes, J. Gu, H. Hagen, B. Hamann, W. Koegler, J. Laurent, J. Meredith, P. Messmer, E. Otoo, V. Perevoztchikov, A. Poskanzer, O. Ruebel, A. Shoshani, A. Sim, K. Stockinger, G. Weber, and W.-M. Zhang. Fastbit: Interactively searching massive data. *J. of Physics: Conference Series*, 180(1), 2009.

[33] K. Wu, E. Otoo, and A. Shoshani. A performance comparison of bitmap indexes. In *Proc. 10th Inter. Conf. on Inf. and Knowl. Manag.*, pages 559–561, 2001.

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