Eurographics Conference on Visualization (EuroVis) 2014
H. Carr, P. Rheingans, and H. Schumann
(Guest Editors)

Volume 33 (2014), Number 3

Sparse Representation and Visualization for Direct Numerical Simulation of Premixed Combustion

Timo Oster[1] Dirk J. Lehmann[1] Gordon Fru[1] Holger Theisel[1] Dominique Thévenin[1]
[1] University of Magdeburg, Germany

Abstract

Direct Numerical Simulations of premixed combustion produce terabytes of raw data, which are prohibitively large to be stored, and have to be analyzed and visualized. A simultaneous and integrated treatment of data storage, data analysis and data visualization is required. For this, we introduce a sparse representation tailored to DNS data which can directly be used for both analysis and visualization. The method is based on the observation that most information is located in narrow-band regions where the chemical reactions take place, but these regions are not well defined. An approach for the visual investigation of feature surfaces of the scalar fields involved in the simulation is shown as a possible application. We demonstrate our approach on multiple real datasets.

Categories and Subject Descriptors (according to ACM CCS): I.6.6 [Simulation and Modeling]: Simulation Output Analysis—

1. Introduction

Direct Numerical Simulation (DNS) is an extremely precise method for simulating turbulent combustion processes. It resolves the underlying equations on a high-resolution grid without any modelling assumptions. A single DNS run produces terabytes of multiple time-dependent 3D scalar fields that have to be stored, analyzed, and visualized. Due to the sheer size of the data, storing it in raw form is prohibitively expensive. This in turn prevents effective analysis and visualization. Existing approaches only focus on a subset of these problems, and fail to use the synergies an integrated approach has to offer. We introduce a sparse data representation tailored to DNS data of premixed combustion that can directly be used for further analysis and visualization, and that retains the possibility of reconstructing the full scalar fields.

Our approach is based on the observation that most of the relevant information is located in narrow-band regions where the chemical reactions take place, the so-called flame fronts. The different scalar fields behave similarly in these regions. In the presence of turbulent flow, the flame front develops complex, wrinkled shapes. The differences between scalar fields affected by turbulence, and their development over time give meaningful insights into the chemical process and its interaction with turbulent flow. An integrated approach for the visual analysis of such DNS data must solve three problems:

1. Handling the *enormous size* of the simulation data
2. *Analyzing* the behavior of the scalar fields
3. *Visualizing* the differences between the scalar fields

The sparse data representation presented here considers all these problems. We fit domain-aware models to the data, exploiting its characteristics and transforming it into a lower-dimensional parameter space that retains all relevant information. In this space, the analysis and visualization can be carried out without accessing the original data. If necessary, the original scalar fields can still be reconstructed.

The paper is organized as follows: In Section 2, we provide background information and present related work. The sparse data representation is introduced in Section 3. Section 4 presents our visual analysis approach, and Section 5 discusses our results and presents feedback from domain experts.

2. Background and Related Work

DNS owes its precision to the lack of any modeling assumptions and the high resolution of the simulation grid. Important DNS applications are presented in [HTERT04]. Most of the simulated variables are related to chemistry. Simulations of practically relevant reactions are complex and involve tens or even hundreds of chemical variables. Due to both the high resolution and the large number of variables, a simulation run produces terabytes of raw data. These amounts cannot be stored or transferred efficiently, prompting the need of methods for either analyzing on-the-fly or reducing them.

Analysis of such raw data is usually done in a post-processing step: Bremer et al. [BWT*09] present a framework

for exploring burning clusters in combustion simulations by generating a level-set-based topological hierarchy; Akiba et al. [AMCH07] display multiple superimposed isosurfaces and direct volume renderings. Further post-processing methods are described in Zistl et al. [ZHJT09]. All approaches rely on the raw data being available after the simulation, thus ignoring problem 1. If the raw data is too large to be stored completely, post-processing can not be performed at all.

To address this issue, on-the-fly visualization methods have been developed. They process and render simulation data while being produced, operating in parallel with the simulation on the same supercomputer. This type of in-situ visualization is discussed in [Ma09], while [YWG*10] describes a specific renderer for volume and particle data. These methods provide a superior way of monitoring the simulation progress. However, they do not offer a solution to problem 2. Since they only output rendered images, more detailed analysis still has to be performed on the huge original data sets.

One way to achieve both flexibility and storage-efficiency is to compress the data before storing it. The compressed data can then be decoded and analyzed with any method. Numerous compression methods exist, but few are well-suited for DNS data. The 3D-SPIHT (Set Partitioning In Hierarchical Trees) algorithm [KXP00] uses a global wavelet transform to compress volume data. A blockwise wavelet-based approach tailored to visualization applications is proposed in [NS01]. Fout et al. [FMA05] describe a vector quantization approach designed for fast decompression using graphics hardware, enabling fast volume rendering of compressed data. These approaches provide high-accuracy, lossy compression for general volume data. However, they do not take into account the special requirements of analyzing combustion data, thus ignoring problems 2 and 3.

Our work focuses on DNS of premixed combustion: generally, a mixture of chemical species react and form products, as described in [FJT11]. Reactions do not happen simultaneously in the whole mixture. Rather, flame fronts spread out from spots of ignition. Reactions only occur in these thin zones, consuming fuel and oxidizer and leaving hot products behind. During the reaction, intermediate, unstable compounds form, which only occur in the flame front and quickly react with other compounds to form stable end products. Outside of the flame front, and therefore in most of the simulated volume, concentrations of chemical variables are close to constant. Inside, these variables vary most rapidly along the shortest path across the front, and their values are therefore highly correlated.

To simplify the investigation of the flame shape, a flame surface representing the flame front is commonly defined as the surface separating burned and unburnt gases. It is contained in the flame front and its surface normal is aligned with the gradients of the scalar variables. Interaction with turbulence causes this flame surface to become wrinkled, producing areas of high curvature. The flame surface can be defined in different ways based on the scalar fields of the simulation. Often it is defined as an iso-surface, surface of maximum value, or surface of steepest slope. These feature surfaces are similar, but their differences give meaningful insights into the combustion process. Our sparse representation directly captures these feature surfaces and enables their analysis and visualization (see Figure 8).

3. A Sparse Representation for DNS Data

To address problem 1, and enable addressing problems 2 and 3, we introduce a sparse representation for DNS data. The transformation into the sparse form consists of three steps: First, points are seeded on the flame surface. Second, the values of the simulation variables are sampled along lines orthogonal to the flame surface, emanating from these points. Third, the resulting profiles are approximated by models, reducing each profile to a set of model parameters.

As mentioned before, simulation variables vary most along the shortest path across the flame front. This direction is approximately orthogonal to the flame surface. We exploit this by representing the 3D scalar fields of the variables as a set of profiles on 1D lines. These profile lines are anchored at points $\mathbf{p}_i$ on the flame surface and oriented along its normal $\mathbf{r}_i$. By giving the lines a limited length, we discard information from outside of the flame front, where the variables are almost constant. The placement of the profile lines is dependent on the flame surface curvature, seeding more lines in highly curved areas and less in planar regions (see Figure 1).

Variables are grouped into three classes: *Reactants* have high concentrations outside of the burnt region and low concentrations inside, *products* are the opposite, and *intermediates* occur near the flame surface between unburnt and burnt regions but have low values on either side. This behavior can be modeled with few degrees of freedom, reducing the amount of data even further. The information that has to be stored in the sparse representation consists of the locations and directions of the profile lines, $\mathbf{p}_i$ and $\mathbf{r}_i$, the model parameters for each variable and profile line, and the full flame surface mesh for subsequent visualization. We now describe how the lines are seeded, how the profiles are extracted from those lines, and how these profiles are then approximated by simple models.

3.1. Strategy for Seeding Profile Lines

Commonly, the flame surface in premixed combustion is defined as the 0.5-isosurface of a *combustion progress variable* [PV12]. This variable is defined as $\frac{T-T_u}{T_b-T_u}$, where T_u and T_b are the temperature of the burnt and unburnt gases, and varies between 0 and 1. The choice of the isovalue is very robust. Our experiments show that variations of ± 0.1 lead to isosurfaces with Hausdorff distances less than 2% of the longest side of the simulated domain. Anchor points $\mathbf{p}_i$ for profile extraction are distributed on this surface, which we extract using CGAL [BO05]. This yields a mesh with approximately uniformly-spaced vertices and edges of approximately the same length as the voxel size in the original data. Since

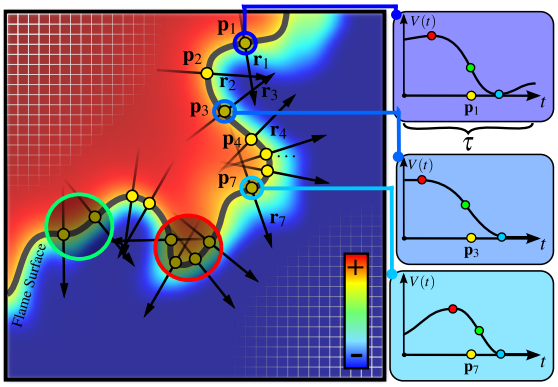

Figure 1: Cross section scheme of flame surface with seeded points $\mathbf{p}$: profiles sampled from profile lines $\mathbf{p}(t)$ at different locations on the flame surface. The minimum, inflection, and maximum of the sigmoidal shape are indicated by the blue, green and red dots. If the sampled line enters or leaves the burning region multiple times, multiple sigmoidal shapes can occur (bottom right box). Seeding density is dependent on the surface curvature (see green vs. red circle).

this results in a large number of vertices, we select only some of them as seed points, using a rejection sampling based on local surface curvature. We estimate the principal curvatures κ_1 and κ_2 at each vertex (see, e.g., [KWTM03]). The curvatures are then transfomed into a seeding density by a logarithmic function:

$$\rho(\kappa_1,\kappa_2) = \frac{\ln(1+\sqrt{\kappa_1^2+\kappa_2^2})}{q} .$$

Here, $q > 1$ steers the seeding density. For each initial vertex, a uniformly distributed random number $r \in (0,1)$ is now generated, and the point is selected if $\rho < r$. This results in few seed points in areas without curvature, and all vertices being selected in areas where $\rho > 1$. Areas of higher curvature get more seeds than less curved ones. Hence, the storage size of the sparse representation depends on the surface shape rather than on the resolution of the data. Adjusting q changes the total number of seed points and balances size vs. quality.

3.2. Extracting Profile Lines

Once the points $\mathbf{p}_i$ are seeded, profiles of all variables are sampled at regular intervals t_j along lines $\mathbf{p}_i(t_j) = \mathbf{p}_i + t_j \cdot \mathbf{r}_i$, with $||\mathbf{r}_i|| = 1$ (Figure 1). Due to the high resolution and accuracy of DNS data, trilinear interpolation is sufficient. The length of the line is chosen sufficiently large to cover the whole reaction zone. The resulting profiles are approximated by simple models to further reduce the required storage space.

3.3. Model-Based Data Approximation

Although the main advantage of DNS is its high precision due to the lack of any modeling assumptions, we use models to describe its outcome in a lower- dimensional form. These models are sufficient to facilitate the analysis of the scalar fields we intend. Additionally, they reduce the space needed to store the sparse representation.

Reactants, intermediates, and products each have very similar profiles that can be locally approximated by simple models. Reactants and products tend to exhibit profiles with a sigmoidal shape, transitioning from a constant high/low value in the unburnt region to a constant low/high value in the burnt region, passing an inflection point in between. This behavior can be expressed by a model based on a sigmoidal function. Intermediate species have a maximum near the flame surface, decreasing on both sides. They are approximated by a model based on a Gaussian bell curve. Small fluctuations that deviate from these models may occur but are not relevant for the general shape. Greater deviations appear if a sample line crosses the flame surface multiple times, which happens if multiple parts of the flame front come close to each other. In such cases, the characteristic behavior occurs multiple times across the profile. To handle this, the instance closest to the anchor point is identified and isolated from the others.

3.3.1. Model for Reactants and Products

Sigmoidal shapes are commonly expressed with the logistic function $1/(1+e^{-x})$, which we extend with parameters γ, adjusting the slope at the inflection and a, determining the limits of the function at positive/negative infinity:

$$\mathfrak{s}(x,a,\gamma) = \frac{2a}{1+e^{(-2\frac{\gamma x}{a})}} - a .$$

While this function can roughly approximate the profiles of reactants and products, it has too few degrees of freedom to reproduce the different curvatures of the profiles at both sides of the inflection point. For this reason, we use two pieces of this function, joining smoothly at the inflection point (Figure 2, top).

$$\mathfrak{S}(x,a_l,a_r,\gamma,x_m,y_m) = \begin{cases} \mathfrak{s}(x,a_l,\gamma)+y_m, & \text{if } x \le x_m \\ \mathfrak{s}(y,a_r,\gamma)+y_m, & \text{if } x > x_m \end{cases}$$

where (x_m,y_m) is the location of the inflection point, γ is the slope at the inflection and $y_m - a_l$ and $y_m + a_r$ are the limits of the function approaching negative and positive infinity.

3.3.2. Model for Intermediate Species

The profiles of intermediates resemble Gaussian bell curves. Since minimum and maximum of the profiles of intermediate species can vary, it is necessary to extend the standard bell curve with additional parameters y_m, the value at the maximum, and y, the limit at infinity.

$$\mathfrak{g}(x,x_m,y_m,\sigma,y) = e^{-\frac{1}{2}(\frac{x-x_m}{\sigma})^2} \cdot (y_m - y) + y .$$

Since the profiles tend to have a steeper slope on the unburnt side and some of them do not reach zero on the burnt side, a two-sided model is once again needed to accurately capture this behavior. We join the two bell curves smoothly at their

maximum point, resulting in a model with six parameters:

$$\mathfrak{G}(x,x_m,y_m,\sigma_l,y_l,\sigma_r,y_r) = \begin{cases} \mathfrak{g}(x,x_m,y_m,\sigma_l,y_l), & \text{if } x \le x_m \\ \mathfrak{g}(x,x_m,y_m,\sigma_r,y_r), & \text{if } x > x_m \end{cases}$$

where (x_m, y_m) is the location of the maximum, σ_l and σ_r determine the slope of the left and right part of the function and y_l and y_r are the limits of the function approaching negative and positive infinity (Figure 2, bottom).

3.3.3. Fitting the Models to the Profiles

We now approximate the profiles by fitting the models. For the sigmoidal model, we need the position, value and first derivative at the inflection point as well as the minimum and maximum values of the profile on both sides of the flame front. For the Gaussian model, the location and value of the maximum near the flame front have to be known, as well as the minimum values y_l and y_r. To robustly determine these feature points, we have to account for the two types of deviations that may occur in the profiles as described above: small fluctuations, and the profile entering and leaving the reaction zone multiple times.

To eliminate small fluctuations, we filter the profiles with a Gaussian kernel [JÖ5]. The kernel size depends on the size of the fluctuations but can be chosen quite small. For the test data (see Section 3.5) we used a size of 6 voxels, which translates to $\sigma = 1$. Extrema are found at zero-crossings of the first derivative of the filtered profile. Possible shifts of the extrema due to the filtering are corrected by mapping each maximum to the largest maximum, and each minimum to the smallest minimum in the radius of the filter size, taking care that extrema do not switch sides. For finding inflection points, we use the same approach but on the second derivative.

Due to the possibility of crossing the flame surface multiple times (see Figures 1 and 2), we can find more feature points on the profile than we need for our models. We have to identify the ones closest to the anchor point. For the sigmoidal model, this is the inflection nearest to the center of the profile. The position and value of this point determine x_m and y_m of the sigmoidal model, while γ is determined by the first derivative at the inflection. The values of the first minimum and maximum left and right of the inflection (depending on the sign of γ) determine a_l and a_r. The positions of these extrema, x_l and x_r, are the boundaries of the portion of the profile that the model is fitted to. The rest of the profile, possibly containing other crossings of the flame surface, is not considered, as those other crossings are already captured by other profile lines seeded there. For the Gaussian model, the maximum nearest to the center determines x_m and y_m, while the values of the closest minima to both sides determine x_l and x_r, as well as y_l and y_r. Further extrema are ignored. For both models, if there are no extrema on either side of x_m, the values at the ends of the profiles are chosen instead.

While the feature points on the profile already determine all parameters for the sigmoidal model, the values for σ_l and

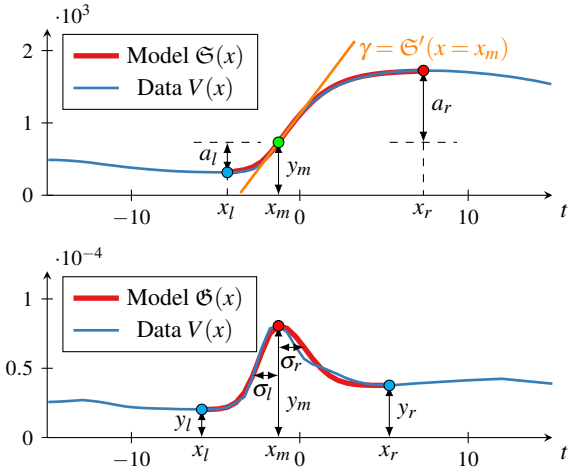

Figure 2: Sigmoidal model (top) and Gaussian model (bottom) with examples of models fitted to a profile.

σ_r of the Gaussian model still have to be found. We obtain an initial guess for $\sigma_{l/r}$ by transforming the intervals of the profile between x_l and x_m and between x_m and x_r into the interval $[0,1]$ and regarding them as halves of two symmetric probability density functions (pdf). The variance of a discrete random variable X with the pdf $p(x)$ and expected value μ is given by: $\mathrm{Var}(X) = \int p(x) \cdot (x-\mu)^2 \, dx$. Since for a normal distribution, the variance is σ^2, we can directly use this equation on our transformed profiles. We then refine this estimate with a simple optimization scheme such as Newton's method to find the optimal fit.

With this, the original data is now described by the flame surface mesh, the $\mathbf{p}_i$ and $\mathbf{r}_i$ of the profile lines, and the model parameters for each profile line and variable. For the sigmoidal model, this is $(a_{li}, a_{ri}, \gamma_i, x_{mi}, y_{mi}, x_{li}, x_{ri})$ for each profile line. For the Gaussian model, the parameters are $(x_{mi}, y_{mi}, \sigma_{li}, y_{li}, \sigma_{ri}, y_{ri}, x_{li}, x_{ri})$. In the following, we describe the reconstruction of the dense scalar fields from this sparse representation.

3.4. Reconstructing the Scalar Fields

If desired, the scalar fields on the original grid can be reconstructed from the sparse representation by interpolation. We sample the fitted models in regular intervals between the respective x_l and x_r of each line and variable. We then apply standard interpolation methods to this set of points to reconstruct the data on the original grid.

We compared two local interpolation methods for scattered data. The first method is a k-approximate-nearest-neighbor (kANN) [AMN*98] interpolation scheme that weighs the values of the k approximate nearest neighbors using Shepard's inverse distance weights. The second interpolation method first generates a tetrahedral mesh from the data points using a Delaunay triangulation. The values inside the mesh cells are then linearly interpolated. This always produces a continuous solution if there are no degenerate mesh cells.

Interpolation methods providing higher smoothness exist. However, these are more computationally expensive and it is not guaranteed that they produce results closer to the original data than the simpler methods.

3.5. Quality and Performance Evaluation

We evaluated the accuracy of the sparse representation on single time steps of three data sets. Each time-step of data sets SYNGAS I and SYNGAS II has 200^3 voxels and 13 variables each. SYNGAS III has 100^3 voxels and 3 Variables. All data sets are from DNS computations of turbulent premixed spherical syngas flames. SYNGAS I contains a flame with strong wrinkles. SYNGAS II has a flame with smaller wrinkles. SYNGAS III contains a flame that has been torn into smaller parts by turbulence.

First, we investigate the error from approximating the original data by our models. We computed the average root mean square (RMS) error between the original data of the profiles and the fitted models. The data values in the range x_{li} and x_{ri} on each profile line are considered. We used normalized RMS errors E^V_{fit} in order to make the variables V comparable:

$$E^V_{\text{fit}} = \frac{(\max(V)-\min(V))^{-1}}{\sum_{i=1}^{n} x_{ri}-x_{li}} \sum_{i=1}^{n} \sum_{\{j|x_{li}\le x_{ji}\le x_{ri}\}} \frac{|P^V_{ij}-u^V_{ij}|}{x_{ri}-x_{li}} .$$

Here, u^V_{ij} is the value of the fitted model corresponding to the original profile value P^V_{ij}, while $\{j|x_{li}\le x_{ji}\le x_{ri}\}$ are the indices of all points on the profile between x_{li} and x_{ri}. We computed this error for all possible profile lines in all data sets. As Figure 3 shows, the errors are quite low, ranging from 0.1% to 6.3% of the respective variable's range.

For investigating the overall error after reconstruction, We computed the reduction ratio c for different q and compared it to the deviation from the original data after reconstruction. The reduction ratio is defined as the storage space needed for the original data divided by the space needed by the sparse representation. We used a normalized root mean square error metric to compute the reconstruction quality:

$$E^V_{\text{reconst}} = \frac{1}{|H|} \int_{x\in H} \left| \frac{R(x)-V(x)}{\max(V)-\min(V)} \right| dx ,$$

where V is the original data of the variable, R is the reconstructed data, H is the convex hull of all points used to reconstruct the data, and $|H|$ is the volume of H. The range of values of variable V is described by $\max(V)-\min(V)$.

We compared the results of linear and kANN interpolation for reconstruction. Our experiments show that for the kANN interpolation a combination of five nearest neighbors weighted with a Shepard weighting function using an exponent of 20 gave the lowest errors. Therefore, we illustrate the results for these parameters only. We also compared our results to the error introduced by naïvely downsampling the data to the same storage size needed by the sparse representation. This is the currently established way of reducing the

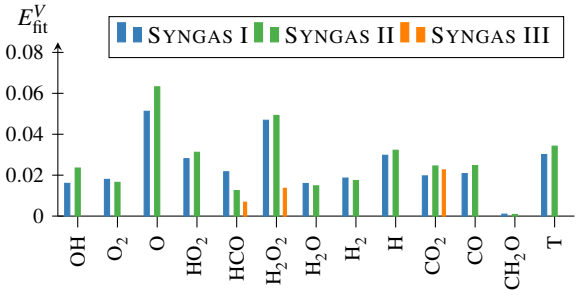

Figure 3: Mean RMS fitting error for all variables. Note that not all data sets contain the same variables.

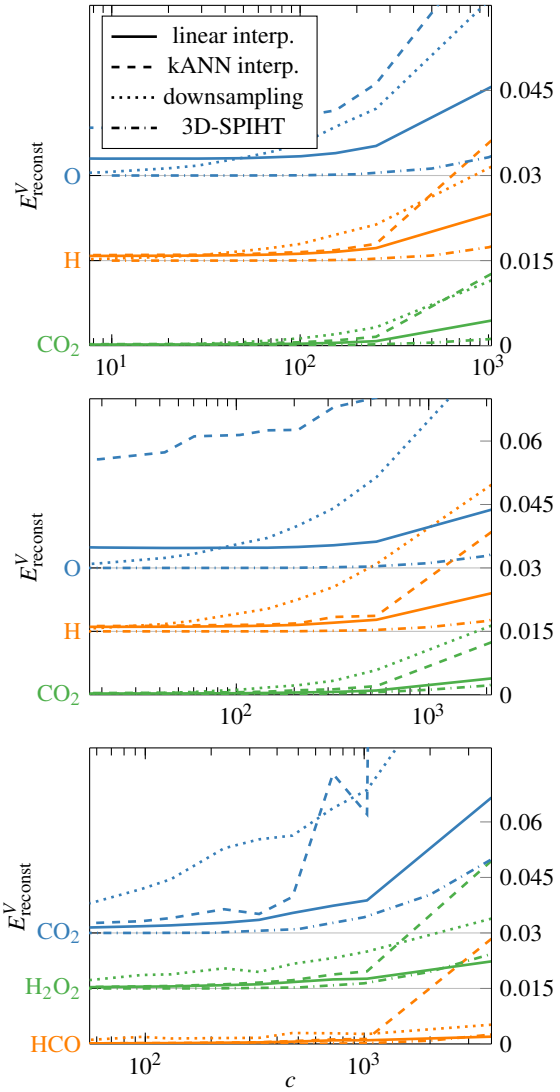

Figure 4: Error vs. reduction ratio for selected variables. Top: SYNGAS I, middle: SYNGAS II, bottom: SYNGAS III. The plots corresponding to each variable are shifted by a constant increment. The horizontal lines signify the zero-levels of the respective plots.

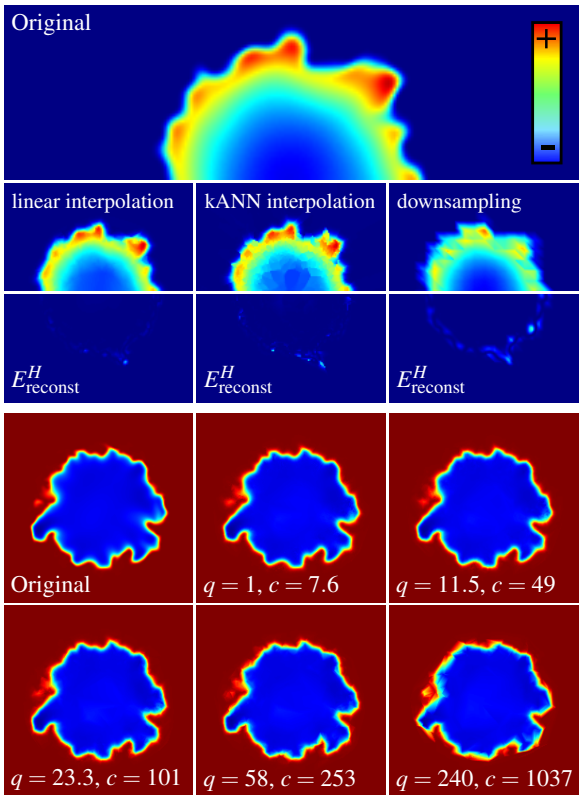

Figure 5: Top: SYNGAS II: Comparison of reconstruction results for H with $q = 36$ (ca. 2500 profile lines, $c = 321$). Bottom: SYNGAS I: Reconstruction results of linear interpolation for different sample densities for O_2.

size of DNS data. For comparison with a dedicated compression algorithm, we used the well-established 3D-SPIHT algorithm [KXP00] implemented in the QccPack library [Fow00] on our data.

Figure 4 shows the reduction ratio c vs. the reconstruction error E^V_{reconst} for all tested methods for selected variables. Please note that for SYNGAS II, higher reduction ratios are achieved than for SYNGAS I, due to the flame in the former being relatively smaller. For small reduction ratios, the downsampling approach naturally performs better, because it does not introduce errors due to model assumptions. For higher reduction ratios, which are needed in practice, our sparse representation always performs significantly better. It is also apparent that linear interpolation performs better than kANN in almost all cases. As a dedicated compression algorithm, 3D-SPIHT achieves better reconstruction quality than our approach. It is however necessary to decompress the data back to its full size before any analysis can be carried out, thus losing a major advantage of our sparse representation, which captures important data features by design.

To further illustrate the performance of our approach, we tested it on eight time steps of the data set HYDROGEN. This data set is from a turbulent premixed spherical hydrogen

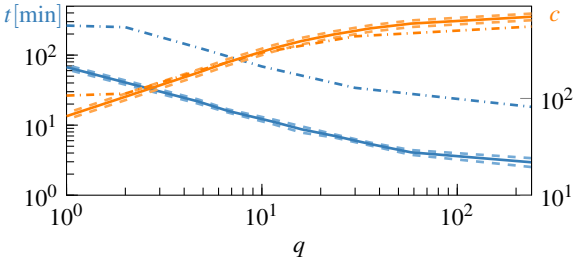

Figure 6: Reduction ratio and computation time for different seeding densities q. Plot shows mean (solid) and standard deviation (dashed) over eight time steps of data set HYDROGEN and values for the synthetic data set (dash-dotted).

flame and has a resolution of 400^3 voxels and 11 variables. This is a typical size for our cooperation partner. One time step amounts to about 5 GB of data. The whole simulation has hundreds of thousands of time steps. Even storing a fraction of them quickly results in terabytes of data. We selected eight time steps from a late (most complex) stage of the simulation. Choosing $q = 5$, which retains very high accuracy for reconstruction, we reduced them to about 30 MB each. This means only 0.6% of the original storage space is required.

For a scalability test, we generated a synthetic data set with 900^3 voxels and 11 variables. This is three times the maximum volume our cooperation partner is able to simulate. We created noise based on an isotropic turbulence frequency spectrum [FE03]. This noise was added to a low frequency component to emulate larger flame structures with smaller surface pertubations. Thresholding produces a surface on which average profiles of the different variables were superimposed. Domain experts confirmed the similarity of the result to real simulations, making it suitable for scalability tests. Since each profile line has to be processed separately, the run time of the algorithm is approximately linear in the number of seed points, and thus depends indirectly on the flame surface area and structure. Figure 6 shows run times and compression ratios for the synthetic data set and HYDROGEN. Data set volume and flame surface area of the synthetic data set are one order of magnitude higher than that of HYDROGEN. This results in a run time which is also one order of magnitude higher, confirming the scalability of our approach.

We have shown that full scalar fields can be reconstructed with good agreement to the original data from a sparse representation requiring a small fraction of the storage space. This representation explicitly captures features such as the points of steepest slope of reactants and products and the point of maximum concentration of intermediate species. In the following section, these feature points are used to visualize the relations between feature surfaces of different variables.

4. Extraction and Visualization of Feature Surfaces

The models used in the sparse representation directly capture the characteristics of the scalar fields in the vicinity of the flame surface. This information can directly be analyzed,

thus solving problem 2 discussed in Section 1. The model parameters x_l, x_m and x_r (see Figure 2) describe three classes of feature points on the profiles: *minimum* (min), *maximum* (max) and *inflection point* (infl). These feature points span feature surfaces that represent alternative definitions of the flame surface (such as the surface of maximum heat release), or inner and outer bounds of the burning region (e.g. minimum and maximum fuel concentration). Investigating the shapes and local distances of these surfaces gives insight into the local combustion process and how it is affected by turbulent flow. We now describe the construction of those feature surfaces and their subsequent visualization.

4.1. Feature Point Construction

As shown in Figure 7 (left), each class of feature points can be transformed into a point set in space. By considering all profile lines $\mathbf{p}_i(t) = \mathbf{p}_i + t \cdot \mathbf{r}_i$, which have been sent out from the flame surface S (Figure 7, left), the position of a feature point $\mathbf{p}(t_f)$ with $f \in \{\min, \max, \text{infl}\}$ is given by

$$\mathbf{p}(t_f) = \mathbf{p} + t_f \cdot \mathbf{r}.$$

Such a feature point set is a discrete sampling of a continuous feature surface S_f. In the following, we discuss our approach of constructing this feature surface from a feature point set.

4.2. Feature Surface Construction

Remember that during the transformation to the sparse representation, an isosurface mesh M representing the flame surface S was extracted. The profile lines were seeded at vertices of this mesh. For each point $\mathbf{p}_i$ on the flame surface we therefore know the position of a point on the feature surface S_f by the corresponding shift value t_{f_i} and the direction of the profile line $\mathbf{r}_i$ (Figure 7 (left)). Hence, the idea is to transform the flame mesh M into a feature mesh M_f, representing the feature surface.

The simplest way to implement this transformation would be to move the vertices $\mathbf{m}_j \in M$ of the flame mesh M along the direction vectors $\mathbf{r}$ of the profile lines to a related feature point, given by the shift value t_f. Unfortunately, the values for t_f and $\mathbf{r}$ are only known at vertices where profile lines have been seeded (see Section 3.1, Figure 7). Thus, the first step of the transformation is to obtain this information for each vertex $\mathbf{m}_j$ of the flame mesh M. We use a diffusion-driven approach to obtain the directions $\mathbf{r}'_j$ and shift values $t'_{f,j}$ for each mesh vertex from the original $\mathbf{r}_i$ and $t_{f,i}$. For this, we fix the original values at the vertices corresponding to points $\mathbf{p}_i$ and diffuse them over the rest of the mesh until convergence. Different diffusion methods can be used to obtain results of varying smoothness. For simplicity, we use an explicit weighted averaging scheme, iteratively replacing the values at each vertex with the sum of its neighbors, weighted with the neighbor's inverse distance. After this process has finished, directions and shift values are known for each vertex $\mathbf{m}_j$ (grey arrows and dots in Figure 7 (right)). This process is a preprocessing step that has to be performed only once before visualization and does not further impede performance.

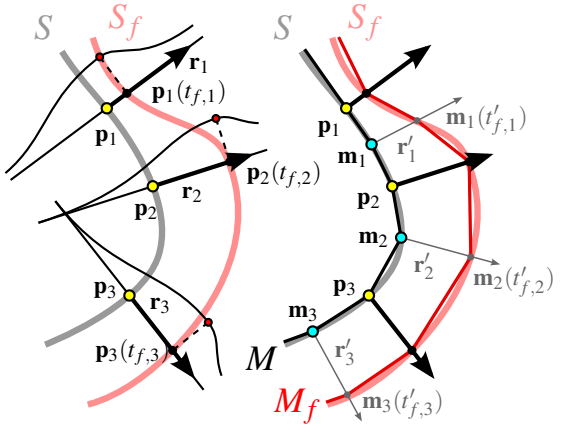

Figure 7: Construction of feature points and feature mesh. Left: Feature points are constructed by shifting the anchor points $\mathbf{p}_i$ on the flame surface S along their direction vectors $\mathbf{r}_i$ by amount t_f taken from the model parameters (here, x_m of a gaussian model). The resulting points $\mathbf{p}_i(t_{f,i})$ are located on feature surface S_f. Right: Directions and model parameters for mesh points $\mathbf{m}_j$ are obtained from points $\mathbf{p}_i$ by diffusion (grey arrows and dots). Feature mesh M_f is then constructed from flame surface mesh M by shifting all vertices along their corresponding directions by their corresponding values of t_f.

4.3. Feature Surface Visualization

Having constructed feature meshes, we can now approach problem 3 of DNS data analysis: visualization. Domain experts want to visually examine the feature meshes, and investigate the differences between feature meshes of different variables or feature point classes. In the following, we introduce our approach for enabling such an analysis task.

4.3.1. Pairwise Distance Visualization

A visualization of distances between feature surfaces of two different variables must allow for quickly identifying regions of small or large distance, as well as the distances' orientation. We achieve this by displaying the local distance between two feature surfaces color-coded on the flame surface mesh M. This mesh serves as a neutral and common base for comparison, which is related to both feature surfaces.

As mentioned, corresponding vertices $\mathbf{m}_f^V$ and $\mathbf{m}_f^W$ of two different feature meshes can be obtained from the vertex $\mathbf{m}$ by shifting it by two different values t_f^V and t_f^W along the local profile direction $\mathbf{r}'$. Thus, the distance between the vertices is simply the difference between the two shift values.

This distance is computed for each vertex and linearly mapped onto a color map. We use a color map adapted to our application: black for values near zero (the meshes intersect), red to yellow for growing positive distances (one mesh is outside of the other locally), and blue to cyan for negative distances (the opposite applies). We introduce parameter u_1 as a scaling factor for adjusting the color contrast and controlling how much of the data is mapped inside the displayed color

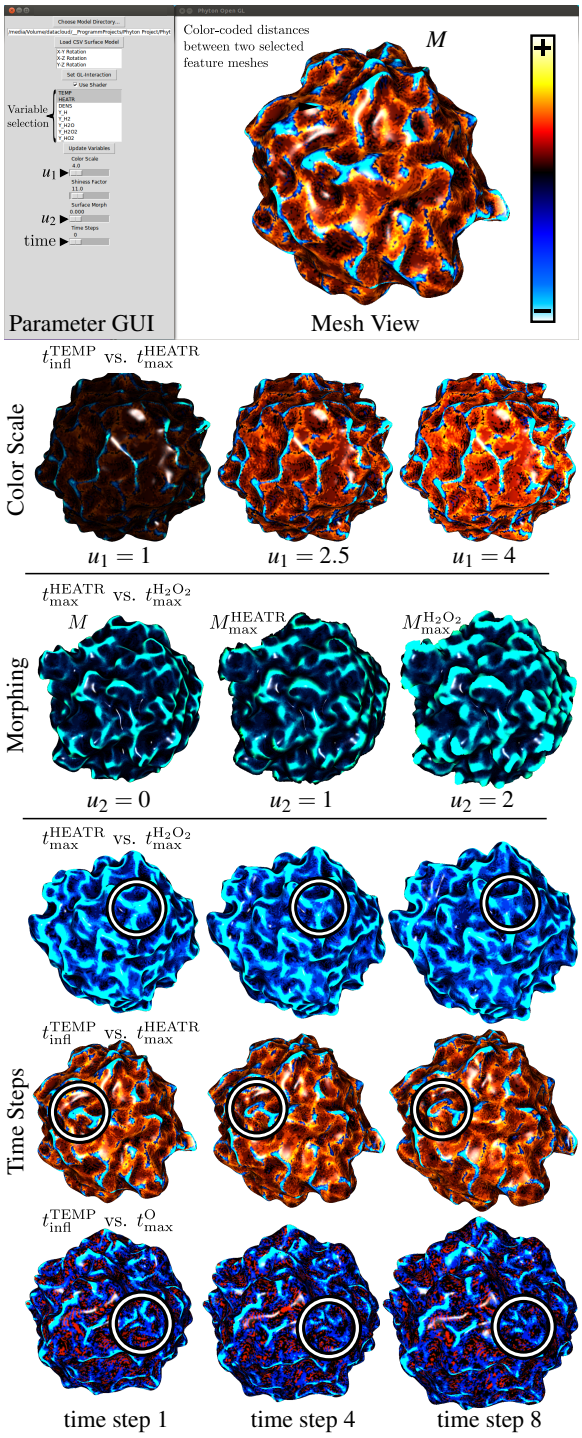

Figure 8: Prototype of visual analysis tool for feature meshes showing dataset HYDROGEN. Top: Graphical user interface. Upper middle: Effect of varying color scale parameter u_1. Lower middle: Morphing between the different surfaces by adjusting u_2. Bottom: Sliding through time to observe the development of the feature meshes (circles).

range and how much is clamped to the maximum/minimum color. This enables a quick visual search for both extreme difference values (by choosing a low value for u_1), or an overview of areas with positive or negative difference values (by choosing a high value for u_1). The color scale and the effect of varying parameter u_1 are shown in Figure 8.

4.3.2. Feature Mesh Visualization

We use standard computer graphics techniques to render the feature surfaces. The distance values are mapped to the mesh as vertex colors, and Phong shading is used to enhance the perception of surface curvature. Larger specular highlights improve the curvature perception but obstruct the view on the mesh color. We therefore let the user control the specular reflectance factor to suit their needs.

To allow for the investigation of the feature meshes' shapes, we provide a user-controlled linear morphing between the flame surface mesh M and the two chosen feature meshes M_f^V and M_f^W. A parameter $u_2 \in [0,2]$ steers the morphing, showing the flame surface M for $u_2 = 0$, the first feature surface M_f^V for $u_2 = 1$ and M_f^W for $u_2 = 2$ (Figure 8). The morphing itself is trivial. Since the corresponding vertices between all the meshes are known and their topology is identical, they just have to be linearly translated as the value of u_2 changes.

Finally, we enable the user to quickly slide through the different time steps and investigate the temporal behavior of the feature surfaces and their relations. This allows for a quick interactive visual analysis that would have been impossible to achieve on the original raw simulation data, due to the large number and storage size of time steps.

4.4. Evaluation of Diffusion Quality

By computing the shift values t_f only for some of the vertices of M, and obtaining them at the other vertices by diffusion, an error is introduced. We quantify this error as the normalized root mean square difference between a ground truth and the values at each vertex after the diffusion process. The ground truth is obtained by computing the model parameters t_f for each vertex of M as described in Section 3.3. The diffusion error $E_{\text{diff}}^{V,f}$ for variable V and feature f is defined as

$$E_{\text{diff}}^{V,f} = \sum_{\mathbf{m}_i \in M} \left| \frac{t_{f,i}^V - t_{f,i}'^V}{\max(t_f^V) - \min(t_f^V)} \right| ,$$

where $t_{f,i}^V$ is the true shift value for variable V and feature f at vertex $\mathbf{m}_i$, $t_{f,i}'^V$ is the corresponding value obtained by diffusion, and $\max(t_f^V) - \min(t_f^V)$ is the range of true shift values over all vertices. We computed this error metric for all variables and time steps of data set HYDROGEN, using different seeding densities q. The results can be seen in Figure 9.

5. Discussion and Conclusion

We introduced a sparse representation for DNS data of premixed combustion. This representation enables an integrated approach for dealing with the three core problems of DNS

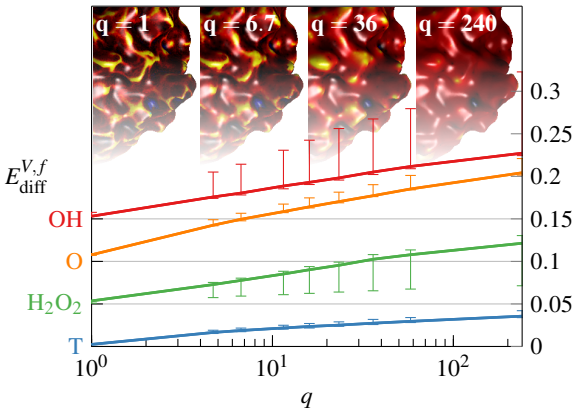

Figure 9: Diffusion error (median, max and min) for selected variables of data set HYDROGEN. Images show qualitative results of visualization for different seeding densities q. The plots corresponding to each variable are shifted by a constant increment.

data analysis (see Section 1). The sparse representation enables storing the simulation results with far smaller space requirements. The space requirement is mainly dependent on the complexity of the flame shape, not on the size or resolution of the data, i.e., the less complex the flame shape, the less profile lines need to be seeded for an accurate representation. Via fitting of models, the sparse representation directly captures important characteristics of the scalar fields that can be analyzed in different ways without the need for data reconstruction. Feature surfaces derived from these models can directly be visualized and facilitate the visual analysis of the data. Previously existing approaches do not address all of the issues mentioned above, restricting the ability to effectively analyze simulation data.

Apart from the feature surfaces, further characteristics might be extracted from the sparse representation, such as gradient fields and their topology. Note that our approach is specifically tailored to DNS data, but can be used for other kinds of multi-field data where changes are only located in narrow-band regions. Such data might be geological data of the material within the earth's mantle, or air field pressure measurements of the supersonic bang of a plane.

Despite its many advantages, our approach has some limitations. Seldom outliers lead to locally large distances of the feature surface to the flame surface, which is visible as spots in the visualization. These are however rare and indicate areas of unusual behavior on the flame surface, which also provides meaningful information. Furthermore, if the flame surface does not capture regions that have non-zero gradients, the information about these regions will be lost. This might happen because, e.g., the temperature there is below the used iso-value. Fortunately, such regions are very rare and solely occur during the short periods of time when a burning region ignites. Thus, practically this case is not relevant. Finally, relying on a random process to seed the profile lines might produce insufficient numbers of samples in some regions. This might be avoided by using a deterministic seeding approach and is left for future work.

Four experts in combustion DNS examined our apprach, two of which were also partly involved in its development. They stated that the extraction of feature surfaces especially for variables that have a maximum near the flame surface is a welcome addition to their set of analysis tools. Such surfaces are of particular interest in the comparison of combustion processes in laminar vs. turbulent flows. Previously, such surfaces were often approximated by iso- surfaces of other variables, which were assumed to be close to the desired feature surface based on the conditions in laminar flow. This hindered the investigation of the effects of turbulence. The possibility of directly comparing feature surfaces with our approach opens new possibilities in the investigation of the effects of turbulence on combustion.

Another application proposed by the experts is the comparison of experimental and simulation research. In experiments, the flame surface is often determined by easily measurable quantities, while more precise definitions are used in simulation research. Our feature surfaces enable comparison of these definitions and deriving models to make experiments and simulations more comparable.

This comparison of different flame surface definitions is also important when comparing different simulations. Since there is no universally agreed-upon definition of the flame surface, different researchers often use different definitions, which could be quantitatively compared with our method.

The sparse data representation, apart from much-needed space savings, opens possibilities of statistical analysis of the relations of feature surfaces, which could be used to improve combustion models for Random Averaged Navier-Stokes or Large Eddy Simulation methods.

Our approach is currently implemented as a post-processing step. By transforming the original data into the sparse representation, disk space usage is reduced considerably. At the same time, the data is also brought into a form better suited for the analysis of the different feature surfaces. In the future, we would like incorporate the approach directly into the simulation and perform the transformation to the sparse representation in-situ.

We have presented a sparse data representation for DNS of premixed combustion that deals with the three core problems of DNS data analysis: Storage requirements, analysis and visualization. The representation accurately approximates the original data while consuming dramatically less storage space. The extraction and visualization of feature surfaces we present as a possible application can be performed without reconstructing the data on the original grid. Experts confirm that our approach opens new and important possibilities of investigating the process of premixed turbulent combustion.

References

[AMCH07] AKIBA H., MA K.-L., CHEN J. H., HAWKES E. R.: Visualizing multivariate volume data from turbulent combustion simulations. *Computing in Science and Engineering 9*, 2 (2007), 76. 2

[AMN*98] ARYA S., MOUNT D. M., NETANYAHU N. S., SILVERMAN R., WU A. Y.: An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM 45*, 6 (Nov 1998), 891–923. 4

[BO05] BOISSONNAT J.-D., OUDOT S.: Provably good sampling and meshing of surfaces. *Graphical Models 67*, 5 (2005), 405–451. 2

[BWT*09] BREMER P.-T., WEBER G. H., TIERNY J., PASCUCCI V., DAY M. S., BELL J. B.: A topological framework for the interactive exploration of large scale turbulent combustion. In *e-Science* (2009), pp. 247–254. 1

[FE03] FERRANTE A., ELGHOBASHI S.: On the physical mechanisms of two-way coupling in particle-laden isotropic turbulence. *Physics of Fluids (1994-present) 15*, 2 (2003), 315–329. 6

[FJT11] FRU G., JANIGA G., THÉVENIN D.: Direct numerical simulation of highly turbulent premixed flames burning methane. In *Direct and Large-Eddy Simulation VIII*, Kuerten H., Geurts B., Armenio V., Fröhlich J., (Eds.), vol. 15 of *ERCOFTAC Series*. Springer Netherlands, 2011, pp. 327–332. 2

[FMA05] FOUT N., MA K.-L., AHRENS J.: Time-varying, multivariate volume data reduction. In *Proceedings of the ACM Symposium on Applied Computing* (2005), ACM, pp. 1224–1230. 2

[Fow00] FOWLER J. E.: Qccpack: An open-source software library for quantization, compression, and coding. In *International Symposium on Optical Science and Technology* (2000), International Society for Optics and Photonics, pp. 294–301. 6

[HTERT04] HILBERT R., TAP F., EL-RABII H., THÉVENIN D.: Impact of detailed chemistry and transport models on turbulent combustion simulations. *Progress in Energy and Combustion Science 30*, 1 (Jan 2004), 61–117. 1

[JÖ5] JÄHNE B.: *Digital Image Processing*. 2005. 4

[KWTM03] KINDLMANN G., WHITAKER R., TASDIZEN T., MÖLLER T.: Curvature-based transfer functions for direct volume rendering: Methods and applications. In *Visualization Conference* (2003), IEEE, pp. 513–520. 3

[KXP00] KIM B.-J., XIONG Z., PEARLMAN W. A.: Low bit-rate scalable video coding with 3-d set partitioning in hierarchical trees (3-d spiht). *IEEE Transactions on Circuits and Systems for Video Technology 10*, 8 (2000), 1374–1387. 2, 6

[Ma09] MA K.-L.: In situ visualization at extreme scale: Challenges and opportunities. *Computer Graphics and Applications 29*, 6 (2009), 14–19. 2

[NS01] NGUYEN K. G., SAUPE D.: Rapid high quality compression of volume data for visualization. *Computer Graphics Forum 20*, 3 (2001), 49–57. 2

[PV12] POINSOT T., VEYNANTE D.: *Theoretical and Numerical Combustion*. 2012. 2

[YWG*10] YU H., WANG C., GROUT R. W., CHEN J. H., MA K.-L.: In situ visualization for large-scale combustion simulations. *Computer Graphics and Applications 30*, 3 (2010), 45–57. 2

[ZHJT09] ZISTL C., HILBERT R., JANIGA G., THÉVENIN D.: Increasing the efficiency of post-processing for turbulent reacting flows. *Computing and Visualization in Science 12*, 8 (2009), 383–395. 2