Eurographics Symposium on Parallel Graphics and Visualization (2017)
J. C. Bennett, A. Telea (Editors)

Interactive Exploration of Dissipation Element Geometry

T. Vierjahn[1,3], A. Schnorr[1,3], B. Weyers[1,3], D. Denker[2,3], I. Wald[4,5], C. Garth[6], T.W. Kuhlen[1,3], and B. Hentschel[1,3]

[1] Visual Computing Institute, RWTH Aachen University, Germany
[2] Institute for Combustion Technology, RWTH Aachen University, Germany
[3] JARA-HPC, Aachen, Germany
[4] SCI Institute, University of Utah, USA
[5] Intel Corporation, USA
[6] University of Kaiserslautern, Germany

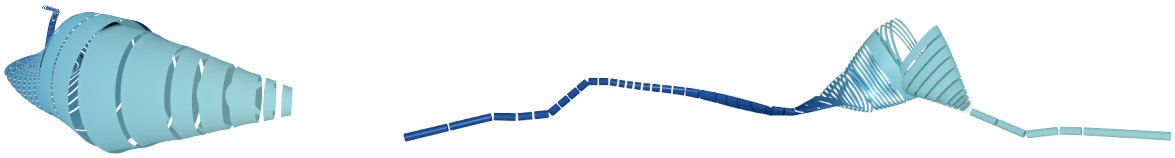

Figure 1: *Approximations for two dissipation elements.*

Abstract

Dissipation elements (DE) define a geometrical structure for the analysis of small-scale turbulence. Existing analyses based on DEs focus on a statistical treatment of large populations of DEs. In this paper, we propose a method for the interactive visualization of the geometrical shape of DE populations. We follow a two-step approach: in a pre-processing step, we approximate individual DEs by tube-like, implicit shapes with elliptical cross sections of varying radii; we then render these approximations by direct ray-casting thereby avoiding the need for costly generation of detailed, explicit geometry for rasterization. Our results demonstrate that the approximation gives a reasonable representation of DE geometries and the rendering performance is suitable for interactive use.

CCS Concepts

•***Computing methodologies*** → ***Scientific visualization; Ray tracing; Parametric curve and surface models;***

1. Introduction

The analysis of large-scale, turbulent flows is a major research field in computational fluid mechanics. In this regard, direct numerical simulation allows domain experts to predict relevant flow quantities across scales. Wang and Peters proposed *dissipation elements* (DEs) as geometric structures for the analysis of small-scale turbulence [WP06, WP08]. Based on an input scalar field, DEs partition the underlying domain into regions of coherent, monotonic gradient flow; specifically, a DE is defined by the set of gradient trajectories which connect a local minimum to a local maximum of the given scalar variable. Given a simulation data set, DEs can be extracted either numerically, i.e. by integrating gradient trajectories [WP06, BSG*11], or via the 3D Morse Smale Complex [EHNP03, GBHP08, GBP12, SN12]. To date, researchers in fluid mechanics have focused on a statistical analysis of DEs in both simulated [AO12, WP06, WP08] and experimental turbulent flows [SDKS11, SGS13].

This paper was motivated by the need to investigate the geometrical structure of dissipation elements in more detail. While a statistical analysis requires only the storage of key quantities – typically the Euclidean distance and the scalar difference between the local extrema – per element, an analysis targeting the geometric structure is far more demanding in terms of storage. The *main goal* of this paper therefore is to devise a method for the compact, memory-efficient description and subsequent visualization of the approximate geometry of dissipation elements – as a first step towards an accurate geometric representation. In order to achieve this goal, we first compute a linear approximation of a DEs centre line along the gradient field. We augment this line by elliptical approximations of

DOI: 10.2312/pgv.20171093

the trajectory point distribution in a cross section plane orthogonal to it (c.f. Section 3). As a second component, we devise a ray casting scheme which directly consumes the previously extracted approximations. Thereby, we avoid the explicit generation of geometry, which saves either computation time or storage space, or both. This rendering stage, which we present in Section 4, relies on the OSPRay framework [WJA*17].

We report our findings in Section 5. These validate that our approximation scheme is efficient compared to the storage of explicit trajectory geometry and lends itself to interactive rendering of approximate local element geometry, yet still depicting the main geometric characteristics of the underlying element data. We derived these findings in close collaboration with domain experts. Thus, they reflect, in part, their professional assessment of the proposed design's suitability for their analysis requirements.

To summarize our *contributions*: we propose a compact representation for DEs; we describe a direct rendering method that avoids geometry generation; we demonstrate our method's effectiveness based on real-world data from direct numerical simulation.

2. Related Work

In this section, we briefly review previous work related to the proposed approximation and rendering schemes. For conciseness, work related to the data being visualized is presented alongside the description of the data itself.

Visualization of Turbulent Flows Flow visualization in general and the visualization of turbulent flows in particular has been a driving use case for scientific visualization at large. Examples include work by Laney et al., who segment the interface surface in turbulent mixing processes based on concepts from topology [LBM*06]; work by Bremer et al., who follow a similar approach in order to analyse the flame surface in turbulent combustion simulations [BWP*10]; and work by Clyne et al. and Treib et al., both of whom target the exploration of large, turbulent flow data sets on desktop machines [CMNR07, TBR*12]. Li et al. use turbulent flow data sets to evaluate a wavelet-based compression scheme [LGP*15].

Icon-based Visualization Icons have been a fundamental modality for flow visualization since the early days of the field [vWPSP96]. More recently, glyphs have been used, e.g., for the visualization of brain scans [Kin04, KW06] and the analysis of blood damage in ventricular assist devices [HTP*08]. Reinders et al. construct a tree-like representation of features by means of a volume thinning algorithm [RJP00] and use the result for feature visualization and tracking. This approach is conceptually similar to our centre line approximation described in the next section.

Ray Tracing for Visualization Though the algorithms described in this paper are general, our particular implementation is based on OSPRay [WJA*17], Embree [WWB*14], and ISPC [PM12].

Embree [WWB*14] is a ray tracing kernel library that offers highly optimized kernels for the fast construction of bounding volume hierarchies; the traversal of rays, packets of rays, and streams of rays through this BVH; and, at least for certain primitive types such as triangles, hair, or subdivision surfaces, the fast ray-primitive intersection.

In addition to those built-in types, Embree also offers so-called *user geometries*, which allow the user to define new geometry types by providing callbacks for computing a primitive's bounding box and ray-primitive intersection, respectively. Given those callbacks Embree then handles BVH construction and ray traversal, calling back to the user-provided callbacks as required.

The Intel® SPMD Program Compiler (ISPC) [PM12] is a "single program multiple data" (SPMD) compiler that allows for writing scalar-looking code which the compiler then automatically maps to the different lanes of modern CPUs' vector units. ISPC offers a C99-style language with some small extensions that, for example, allow a programmer to specify which variables should exist per vector lane (`varying`) vs. per scalar CPU thread (`uniform`).

ISPC supports all modern CPU instruction sets from Intel Streaming SIMD Extensions (SSE) all the way to the AVX512 instructions found on the latest Intel® Xeon Phi™ processors. Most importantly, ISPC generates linkable Object code that can be called from, and can callback into, regular C/C++ code. This allows, for example, to use ISPC code to be interfaced with Embree, using ISPC-based ray primitive intersections in Embree user geometries, etc.

Finally, OSPRay [WJA*17] is a ray tracing based rendering framework for scientific visualization. It uses a object-oriented architecture where different "actors" such as cameras, renderers, geometries, volumes, etc., interact. OSPRay comes with a set of predefined actor types such as different camera types, triangle meshes, spheres, stream lines, etc. In addition, OSPRay offers an SDK that allows users to define additional actor types (e.g., a new geometry type such as our elliptic tubelets) in the form of demand-loadable "modules", which offers users an easy way of adding new actor types that then work together with the rest of OSPRay.

Internally OSPRay builds on top of Embree and ISPC: performance-critical code is written in ISPC, and acceleration data structure construction and traversal are handled through Embree, using either Embree primitives where possible (triangles), or through Embree user geometries.

In using ray tracing for interactive visualization OSPRay follows previous work by Parker et al. [PSL*98, PPL*99], Wald et al. [WSBW01, WBS02], Bigler et al. [BSP06], Brownlee et al. [BPL*12], etc. Ray tracing in particular has benefitted from ever faster hardware, and from the emergence of highly tuned ray tracing libraries such as Embree [WWB*14] and OptiX [PBD*10]: today, a single laptop or workstation can typically deliver sufficient performance to interactively ray-trace non-trivial data sets [WJA*17].

One of the advantages of ray tracing has always been that it can represent many different primitive types without the need for explicit tesselation. VMD's "Tachyon" ray tracer, for example, natively supports cylinders, spheres, and cones to represent particles, bonds, and balls-and-stick models. OSPRay does the same, and also added a piece-wise-linear, constant-radius "stream line" in which control points are connected through cylinders, using spheres at the joints to round the edges. The "Brayns" project [Fav16] expands

on this work by allowing control points with different radius, connected through cylinder segments.

Thick curves are also used in professional rendering, often to represent hair or fur. A method for intersecting the Renderman "curves" primitive is, for example, given by Nakamaru et al. [NO02]. Support for hair primitives was also added to Embree [WBW*14]; this initially used "flat" ribbons for the hair intersection, but has since been extended to more generalized cubic Bezier curves. However, all these approaches only allow circular cross-sections.

In this paper, we extend the use of ray tracing to the immediate rendering of tubelets with elliptic cross section.

3. Dissipation Element Approximation

Our algorithm for the approximation of dissipation elements comprises two main steps: based on a set of input trajectories, which give a high resolution representation of a single DE's geometry, we first compute an approximate centre line; at each of its vertices, we then approximate the element's cross-section orthogonal to the centre line by a 2D ellipse.

For both steps, we decided to use straightforward approaches in order to evaluate the efficacy of the whole approximation-rendering-system. Consequently, there is no guarantee that the approximated centre line lies completely in the interior of a DE. Furthermore, the shape of the cross-section is underestimated by using the variance of the trajectory-distribution instead of, e.g., the convex hull. This will later help separate individual DEs although they are space-filling. Nevertheless, more elaborate approaches are deliberately left for future work.

3.1. Dissipation Element Extraction

Extracting dissipation elements of a locally smoothed scalar field from a single time step is implemented as a two-step algorithm [WP08]. First, trajectories are numerically integrated from each grid point in ascending and descending gradient direction. A special numerical treatment near critical points that is valid for diffusion-controlled scalar fields ensures that trajectories resist small perturbations. When a trace is successfully extended to a pair of local extrema, this trace's seed point is labelled by that pair. In the second step, all grid points, which are assigned to the same pair of extrema, receive a unique DE ID. This algorithm has been parallelized using OpenMP and tuned for large NUMA systems [BSG*11]. For our approximation, we get the set of trajectories as input that make up a single DE. Our current implementation relies on this input representation in two key steps. However, we will point out how it can be adapted to other input formats – e.g. a labelling of grid cells resulting from an Morse-Smale-Complex-based data decomposition.

3.2. Adaptive Centre Line Approximation

Assume the input trajectory set is given as a set of pairs

$$\mathcal{G} = \{(g_i, \phi_i)\} \quad , \text{ with } 1 \leq i \leq N,$$

where the $g_i : \mathbb{R} \mapsto \mathbb{R}^3$ define the trajectories, and the $\phi_i : \mathbb{R} \mapsto \mathbb{R}$ yield the scalar value at each trajectory point. For the sake of simplicity, we assume that the scalar field is normalized to the unit interval for the given DE.

We approximate the centre line of a DE by a series of centres of gravity of isopoints. We define the set of isopoints for a given scalar value σ as

$$\mathcal{P}_\sigma = \{g_i(\alpha) \mid \phi_i(\alpha) = \sigma; 1 \leq i \leq N\} \quad .$$

Then, its centre of gravity is given by

$$\mathbf{c}_\sigma = \frac{1}{N} \sum_{\mathbf{p} \in \mathcal{P}_\sigma} \mathbf{p} \quad .$$

We construct our initial DE centre line approximation from a fixed set of these points; specifically, we chose the points $\mathbf{c}_\sigma$ for $\sigma \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$.

We then refine this approximation by iteratively checking for each line segment whether or not it needs to be refined, and by splitting it as needed. Assume $\mathbf{c}_\sigma$ and $\mathbf{c}_\tau$ to be two successive points in the approximation. We compute the centre of gravity $\mathbf{c}_\mu$ of the isopoints at the intermediate level $\mu = \frac{\sigma+\tau}{2}$. We refine the segment $(\mathbf{c}_\sigma, \mathbf{c}_\tau)$ by inserting $\mathbf{c}_\mu$ iff

$$\left\| \mathbf{c}_\mu - \frac{\mathbf{c}_\sigma + \mathbf{c}_\tau}{2} \right\| > \delta_e$$

for a user-specified error δ_e. Whenever a segment is refined, the iteration is resumed at its original start point $\mathbf{c}_\sigma$. In this way, we effectively implement a depth-first refinement pattern which will terminate as soon as the error constraint is respected for each level. While the above formulation assumes that the DE is approximated by a set of gradient trajectories, the isopoint sets $\mathcal{P}_\sigma$ can be straightforwardly computed from a volume representation of the DE, e.g., one extracted from an Morse-Smale-Complex-based labelling of the data set.

3.3. Cross-Section Approximation

For each centre line segment $(\mathbf{c}_\sigma, \mathbf{c}_\tau)$, we first define its tangent direction as

$$\mathbf{t}_\sigma = \frac{\mathbf{c}_\tau - \mathbf{c}_\sigma}{\|\mathbf{c}_\tau - \mathbf{c}_\sigma\|} \quad . \tag{1}$$

The pair $(\mathbf{c}_\sigma, \mathbf{t}_\sigma)$ defines the plane for which

$$(\mathbf{c}_\sigma - \mathbf{x})^\top \mathbf{t}_\sigma = 0 \quad .$$

We then use this plane as the basis for a cross section through the element centred at $\mathbf{c}_\sigma$.

For each gradient trajectory, the set

$$\mathcal{S}_{\sigma,i} = \left\{ g_i(\alpha) \;\middle|\; (\mathbf{c}_\sigma - g_i(\alpha))^\top \mathbf{t}_\sigma = 0 \right\} \quad , \text{ with } 1 \leq i \leq N,$$

comprises *all* intersections of g_i and the cross section plane at $\mathbf{c}_\sigma$. In order to consider only those intersections that are approximating $\mathcal{P}_\sigma$, we define the set of cross section points as the union of points from $\mathcal{S}_{\sigma,i}, 1 \leq i \leq N$ with minimal distance to the centre:

$$\mathcal{S}_\sigma = \bigcup_{i=0}^{N} \{ \arg\min_{\mathbf{x} \in \mathcal{S}_{\sigma,i}} \|\mathbf{x} - \mathbf{c}_\sigma\| \}$$

This definition of $\mathcal{S}_\sigma$ again assumes a set of trajectories as input. Nonetheless, it is obvious how to transfer $\mathcal{S}_\sigma$ to a grid-based representation of the input dissipation element.

Afterwards, we fit a 2D ellipse, which resides inside the cross section plane, to $\mathcal{S}_\sigma$. To this end, we first establish an arbitrary coordinate frame inside that plane. In keeping with the notations of Frenet-Serret frames, we will call the two basis vectors the normal $\mathbf{n}_\sigma$ and binormal $\mathbf{b}_\sigma$. We set

$$\mathbf{n}_\sigma = \frac{\mathbf{p}-\mathbf{c}_\sigma}{\|\mathbf{p}-\mathbf{c}_\sigma\|}$$

for an arbitrary, off-centre point $\mathbf{p} \in \mathcal{S}_\sigma$ and consequently

$$\mathbf{b}_\sigma = \frac{\mathbf{t}_\sigma \times \mathbf{n}_\sigma}{\|\mathbf{t}_\sigma \times \mathbf{n}_\sigma\|} \quad .$$

Let $\mathbf{p}_j = (\alpha_j, \beta_j) \in \mathbb{R}^2$ denote the 2D coordinate vectors of the intersection points in $\mathcal{S}_\sigma$, such that

$$\forall \mathbf{p}_j \in \mathcal{S}_\sigma \; : \; \mathbf{c}_\sigma + \alpha_j \mathbf{n}_\sigma + \beta_j \mathbf{b}_\sigma \; = \; \mathbf{p}_j \quad .$$

We finally obtain our elliptic approximation by computing the covariance matrix of the resulting points

$$\mathbf{\Sigma}\big(\{\mathbf{p}_j\}\big) = \begin{bmatrix} \operatorname{var}\alpha_j & \operatorname{cov}\alpha_j,\beta_j \\ \operatorname{cov}\alpha_j,\beta_j & \operatorname{var}\beta_j \end{bmatrix}$$

and solving for its eigenvalues $\lambda_{1,2}$ and corresponding eigenvectors $\mathbf{q}_1, \mathbf{q}_2$. Since $\mathbf{\Sigma}\big(\{\mathbf{p}_j\}\big)$ is 2×2, there exists a closed-form solution to this problem. Hence, our implementation avoids a (costly) numerical iteration scheme. Assuming $\lambda_1 \geq \lambda_2 \geq 0$, then $\mathbf{q}_1$ gives the major axis of our elliptical fit and we scale its major radius to $r_1 = \sqrt{\lambda_1}$. Analogously, we set $r_2 = \sqrt{\lambda_2}$ for the ellipse's minor axis given by $\mathbf{q}_2$.

In summary, our DE approximation is an ordered set of sample tuples, each of which has the form $(\mathbf{c}_\sigma, \mathbf{q}_1, \mathbf{q}_2, r_1, r_2)$.

4. Ray Tracing Elliptical Tubelets

Our algorithm renders the approximate DE geometry as a set of straight tube segments, tubelets for short, each with an elliptical profile. It supports ellipses with arbitrary rotation around the direction of the tubelets, and it handles arbitrary non-zero radii. Both, rotation and radii, are allowed to change along the tubelet. In this section, we derive the intersection test between the traced rays and a tubelet's surface. More details leading to our implementation are provided in the appendix in order to help reproduce our findings.

If the ellipse's radii and rotation change too much along a tubelet, a "candy-wrapper" effect will become noticeable. To reduce this effect, we further subdivide such tubelets. Considering this subdivision already during a DE's approximation is left for future work.

Furthermore, the individual tubelets will necessarily intersect each other, whenever a DE's centre line bends. To mitigate this effect we slightly shorten the tubelets, introducing larger gaps between them. Avoiding the intersections and closing the gaps by properly joining the tubelets is left for future work.

4.1. Ray-Tubelet-Intersection

Let $\mathbf{c}_\sigma \in \mathbb{R}^3$ denote the beginning of a tubelet's centre line $\mathbf{c}$, and let $\mathbf{c}_\tau \in \mathbb{R}^3$ denote its end. Then,

$$\mathbf{c}(u) = \mathbf{c}_\sigma + u \cdot (\mathbf{c}_\tau - \mathbf{c}_\sigma) \quad , \quad u \in [0,1]$$

denotes points on that line. Using the tangent direction $\mathbf{t}_\sigma$ from Equation 1 and the reparameterization

$$v \coloneqq u \cdot \|\mathbf{c}_\tau - \mathbf{c}_\sigma\| \tag{2}$$

yields the line's equation

$$\mathbf{c}(v) = \mathbf{c}_\sigma + v \cdot \mathbf{t}_\sigma \quad , \quad v \in \big[0, \|\mathbf{c}_\tau - \mathbf{c}_\sigma\|\big] \tag{3}$$

parameterized by length.

Let $\mathbf{o} \in \mathbb{R}^3$ denote the origin of a ray $\mathbf{r}$, and let $\mathbf{d} \in \mathbb{R}^3$ denote its direction. Then,

$$\mathbf{r}(t) = \mathbf{o} + t \cdot \mathbf{d} \quad , \quad 0 < t \in \mathbb{R} \quad , \quad \|\mathbf{d}\| = 1$$

denotes points on that ray.

Assume there exists a plane perpendicular to the line $\mathbf{c}$ being swept along that line. Then,

$$\mathbf{t}_\sigma^\top \big(\mathbf{x} - \mathbf{c}(v)\big) = 0$$

holds for all points $\mathbf{x} \in \mathbb{R}^3$ in the plane located at $\mathbf{c}(v)$. Intersecting the ray $\mathbf{r}$ with that swept plane yields

$$\begin{aligned} & \mathbf{t}_\sigma^\top \big(\mathbf{r}(t) - \mathbf{c}(v)\big) = 0 \\ \equiv \quad & \mathbf{t}_\sigma^\top (\mathbf{o} + t \cdot \mathbf{d} - \mathbf{c}_\sigma - v \cdot \mathbf{t}_\sigma) = 0 \\ \iff \quad & t \cdot \mathbf{t}_\sigma^\top \mathbf{d} + \mathbf{t}_\sigma^\top (\mathbf{o} - \mathbf{c}_\sigma) = v \quad , \end{aligned}$$

the line's parameter v for the perpendicular plane located at $\mathbf{c}(v)$ containing a given $\mathbf{r}(t)$. Finally, substituting using $e, f \in \mathbb{R}$ for convenience yields

$$t \cdot e + f = v \quad . \tag{4}$$

Assume furthermore there exists an ellipsoid being swept along the line $\mathbf{c}$, and let $\mathbf{A}$ denote a matrix specifying that ellipsoid's orientation and dimensions, then

$$\big(\mathbf{p}_e - \mathbf{c}(v)\big)^\top \cdot \mathbf{A} \cdot \big(\mathbf{p}_e - \mathbf{c}(v)\big) = 1$$

holds for all points $\mathbf{p}_e \in \mathbb{R}^3$ on the ellipsoid located at $\mathbf{c}(v)$. Intersecting the ray $\mathbf{r}$ with that ellipsoid yields

$$\big(\mathbf{r}(t) - \mathbf{c}(v)\big)^\top \cdot \mathbf{A} \cdot \big(\mathbf{r}(t) - \mathbf{c}(v)\big) = 1 \quad . \tag{5}$$

In order to render only the hull of the tubelet, the ray needs to be simultaneously intersected with the swept ellipsoid *and* the swept plane. Consequently, inserting v from Equation 4 into Equation 3 yields

$$\begin{aligned} \mathbf{c}(v) &= \mathbf{c}_\sigma + (t \cdot e + f) \cdot \mathbf{t}_\sigma \\ &= \mathbf{c}_\sigma + f \cdot \mathbf{t}_\sigma + t \cdot e \cdot \mathbf{t}_\sigma =: \mathbf{c}(t) \quad , \end{aligned}$$

turning Equation 5 into

$$\big(\mathbf{r}(t) - \mathbf{c}(t)\big)^\top \cdot \mathbf{A} \cdot \big(\mathbf{r}(t) - \mathbf{c}(t)\big) = 1$$

which is quadratic in t.

4.2. Varying Orientation and Radii

Matrix $\mathbf{A}$ specifies the orientation and radii of the tubelet's elliptical profile. Its eigen decomposition yields

$$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top \quad , \quad \text{where}$$
$$\mathbf{Q} = \begin{bmatrix} | & | & | \\ \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3 \\ | & | & | \end{bmatrix} \quad \text{and} \quad \mathbf{\Lambda} = \begin{bmatrix} r_1^{-2} & 0 & 0 \\ 0 & r_2^{-2} & 0 \\ 0 & 0 & r_3^{-2} \end{bmatrix} \quad ,$$

with orthogonal unit vectors $\mathbf{q}_i, i = 1,\dots,3$ spanning the ellipsoid, and with $r_i, i = 1,\dots,3$ denoting its radii. By definition, in the proposed algorithm

$$\mathbf{q}_3 := \mathbf{t}_\sigma \quad \text{and} \quad r_3 := 1 \quad ,$$

since only the ellipse in the swept plane is required for creating the tubelet's hull but $\mathbf{A}$ needs to be compatible with 3D vectors for computing ray intersections.

Our DE approximation specifies only an $\mathbf{A}_\sigma$ for each tubelet's beginning $\mathbf{c}_\sigma$ (c.f. Section 3). The $\mathbf{A}_\tau$ at the current tubelet's end $\mathbf{c}_\tau$ is computed by projecting the orientation of the next tubelet's beginning into the cross section plane at the current tubelet's end. Consequently, each tubelet has parallel caps. The radii of the current tubelet's end match the radii of the next tubelet's beginning.

In order to let the profile, i.e., the ellipse's orientation and radii, change along the tubelet, we linearly interpolate between an $\mathbf{A}_\sigma$ at the tubelet's beginning and an $\mathbf{A}_\tau$ at the tubelet's end, yielding

$$\mathbf{A}(u) = \mathbf{A}_\sigma + u \cdot (\mathbf{A}_\tau - \mathbf{A}_\sigma) \quad .$$

Using the reparameterization from Equation 2 and v from Equation 4 turns this into

$$\begin{aligned} \mathbf{A}(v) &= \mathbf{A}_0 + v \cdot \frac{\mathbf{A}_1 - \mathbf{A}_0}{\|\mathbf{c}_\tau - \mathbf{c}_\sigma\|} \\ &= \mathbf{A}_0 + (t \cdot e + f) \cdot \frac{\mathbf{A}_1 - \mathbf{A}_0}{\|\mathbf{c}_\tau - \mathbf{c}_\sigma\|} \\ &= \mathbf{A}_0 + f \cdot \frac{\mathbf{A}_1 - \mathbf{A}_0}{\|\mathbf{c}_\tau - \mathbf{c}_\sigma\|} + t \cdot e \cdot \frac{\mathbf{A}_1 - \mathbf{A}_0}{\|\mathbf{c}_\tau - \mathbf{c}_\sigma\|} =: \mathbf{A}(t) \quad . \end{aligned} \tag{6}$$

Interpolating $\mathbf{Q}$ via SLERP and interpolating each r_i linearly would undoubtedly lead to a more accurate representation of the tubelet's shape. However, this would introduce higher-order polynomials to the intersection tests for which root finding gets costly. In contrast, linearly interpolating the $\mathbf{A}_i$ enables us to pose the overall ray-tubelet-intersection as

$$\big(\mathbf{r}(t) - \mathbf{c}(t)\big)^\top \cdot \mathbf{A}(t) \cdot \big(\mathbf{r}(t) - \mathbf{c}(t)\big) = 1$$

which is cubic in t. We use Blinn's numerically robust algorithm [Bli07] to solve this equation in the intersection test.

4.3. Subdividing and Shortening Tubelets

In order to still provide reasonable accuracy, we subdivide individual tubelets on import whenever interpolating the $\mathbf{A}_i$ is likely to introduce too large errors. Whenever the ellipsoid's radii or orientation change too much along a tubelet, we separately compute interpolated $\mathbf{q}_i$ and r_i for the centre of that tubelet. A radius-ratio of 1.5 between start and end of a tubelet, and a basis-rotation of $\frac{\pi}{16}$ turned out to be reasonable thresholds providing results appropriate for the described use case.

Without additional measures, the individual tubelets might intersect each other since our algorithm performs a piece-wise linear approximation of the overall tube. In order to reduce these intersection on small-angle bends, we slightly reduce the length of each tubelet. For this purpose we limit the valid range of the parameter v such that

$$v \in \big[\varepsilon \cdot \|\mathbf{c}_\tau - \mathbf{c}_\sigma\|, (1-\varepsilon) \cdot \|\mathbf{c}_\tau - \mathbf{c}_\sigma\|\big] \quad .$$

$\varepsilon = 0.1$ turned out to produce reasonable results.

5. Results

This section summarizes the results that were obtained using our proposed methods for DE approximation and rendering.

5.1. Test Data

We base our experiments on real-world data from ongoing research towards a deeper understanding of turbulent flows in general and turbulent phenomena in combustion processes in particular [GSNP13, WP13].

premixed jet flame 1 and 2 – The first and second data sets are from of a series of direct numerical simulations of planar spatial evolving premixed turbulent jet flames [LAB16]. These simulations were parametrised to investigate fundamental assumptions of regimes in turbulent premixed combustion. The full computational grids consist of $720 \times 480 \times 256$ and $2880 \times 1920 \times 512$ grid points for premixed jet flame 1 and 2, respectively. The DE analysis was performed on the temperature fields T, where a subdomain was chosen in axial flame position to assure comparability of the DEs. Table 1 refers to the respective subdomains.

non-premixed jet flame – The third data set was generated by a direct numerical simulation of a highly turbulent, temporally evolving non-premixed planar jet flame for the purpose of a detailed investigation of small scale turbulence / chemistry interaction. The fuel consists of diluted methane to incite local extinction and re-ignition for further investigation. The computational grid features $768 \times 1024 \times 512$ grid points with a total of approximately 15 billion degrees of freedom. Including mass fractions of all chemical species involved, diffusion coefficients, local pressure, and velocity fields, roughly 360 GB of total data is generated for every time step of the simulation. A subdomain of the mixture fraction field Z of the direct numerical simulation is chosen for DE analysis, as it provides the most insight into the turbulence / chemistry interaction.

Based on a domain expert's analysis use case, only DEs that crossed a certain isosurface were extracted. For the premixed jet flame 1 and 2 cases, the isothermal contour for $T = 1800\,\mathrm{K}$ was used. In case of the planar jet flame, we selected the elements by means of the mixture fraction isosurface for $Z = 0.2$.

The input data dimensions, sizes – given for a 64-bit, double precision point scalar variable – and resulting DE populations are summarized at the top of Table 1.

Table 1: *Overview of data sets and approximation sizes.*

	jet flame1	jet flame2	non-premixed jet flame
input data			
data res.	$256 \times 320 \times 256$	$512 \times 640 \times 387$	$768 \times 512 \times 512$
size [MB]	160	1,280	1,536
#elements	1,277	8,307	19,341
full geometry representation			
#points	25,273,718	92,185,009	458,309,403
#lines	190,933	756,081	2,245,738
size [MB]	387	1,413	7,010
approx. $\delta_e = 5.0$			
#ellipsoids	18,995	118,753	466,438
avg. line len.	14.87	14.28	24.00
size [MB]	1.16	7.24	28.47
ratio to full	333×	195×	246×
approx. $\delta_e = 10.0$			
#ellipsoids	11,587	70,584	375,744
avg. line len.	9.07	8.49	19,34
size [MB]	0.71	4.21	22.93
ratio to full	545×	336×	306×
approx. $\delta_e = 20.0$			
#ellipsoids	8,114	49,638	262,610
avg. line len.	6.35	5.97	13.52
size [MB]	0.5	3.02	16.03
ratio to full	774×	468×	437×

Table 2: *Data set sizes and rendering performance.*

	Size		Size		Render Time	
Elem. ID	Traject. [MB]	δ_e	Ellipses [kB]	Tubes [kB]	BW [ms]	KNL [ms]
173,987	2.72	5	0.938	4.22	2.7	2.4
		10	0.562	3.09	2.67	2.62
		20	0.328	3.0	2.71	2.77
396,760	2.34	5	1.41	6.28	2.49	3.3
		10	1.17	5.81	2.47	3.26
		20	1.03	4.69	2.37	3.02
581,114	1.85	5	1.31	6.47	3.47	3.22
		10	1.17	6.0	3.44	3.37
		20	0.844	5.44	3.42	3.32
702,625	0.454	5	1.12	4.69	4.32	3.94
		10	0.984	4.41	4.32	3.13
		20	0.797	3.38	4.28	3.72

5.2. Dissipation Element Approximation

Table 1 also summarizes the characteristics of the results of our approximation scheme in terms of space consumption for different error thresholds δ_e. Here, δ_e is given in cell diameters, i.e., $\delta_e = 5$ allows for a maximum approximation error of 5 grid cells (c.f. Section 3). Before the development of our approximation scheme, our collaborators would have to collect and store all trajectories assigned to a particular DE in order to be able to analyse its geometry. With our approximation scheme, the required storage space can significantly be reduced, i.e., by factors between 195× and 774×, depending on the data set and algorithm settings.

Here, we would like to note that we specifically compute these ratios against the size of the full trajectory geometry data that serves as input to our algorithm. This is, indeed, a very detailed yet lavish representation. In Section 3, we indicated how our algorithm could be adapted to a volume-based input representation of element geometries. This would essentially require a single, 64-bit integer label plus the input scalar data. However, since the DEs in our example cases do not cover the entire domain, the exact storage requirements are hard to compute. Therefore, we refrain from adding similar ratios for this representation, but note that they would be significantly lower than the ratios given in Table 1.

Regarding our approximation scheme, We note that the output size does not scale linearly with the error bound; quite to the contrary, reducing the allowable error by a factor of 4 does not quite double the number of output ellipses and thus the overall output size in case of the planar jet flame. We reason that this is due to the fact that the chosen maximum threshold of $\delta_e = 20$ already leads to approximations that capture all essential features well. Further decreasing the error rate therefore helps to adapt the centre line approximation to relatively small-scale features of the respective DEs.

This argument is supported by a visual inspection of a number of DE geometries and their approximations. Figure 2 shows a side-by-side comparison of four selected elements, each of which is given for all four error thresholds plus a depiction of the fully detailed trajectory geometry. As can be seen, the general shape can clearly be discerned from the lowest-resolution approximation, already. Increasing the accuracy by lowering the error threshold allows our algorithm to better capture fine details, specifically in the long tails of some elements, as is evident from the thin tail to the left of element 173,987 and the lower end of 396,760. As discussed in Section 3, our centre-line approximation is not yet guaranteed to lie completely in the interior of a DE, and our approximation underestimates the input point set.

Second, Figure 2 shows that the adaptive centre line refinement scheme places additional points in areas with high variation in the underlying point set. This leads to shorter tubelets in areas of high curvature, whereas mostly linear sections are approximated by longer, linear tubelets. Thus, the tubelet length provides an additional channel to communicate information about the underlying point distribution. The behaviour is clearly evident for the visualization of element 396,760: its body, which seems to twist around the up direction, is depicted by several smaller ellipses; in contrast, both mostly linear tails are visualized by thin, prolonged tubelets.

5.3. Rendering Performance

Performance measurements were conducted on

- a dual-socket node with 2× Intel® Xeon® E5-2680 v4 [Broadwell] processor (BW) featuring 14 cores each at 2.4 GHz, and 128 GB DDR4-2400 RAM, and

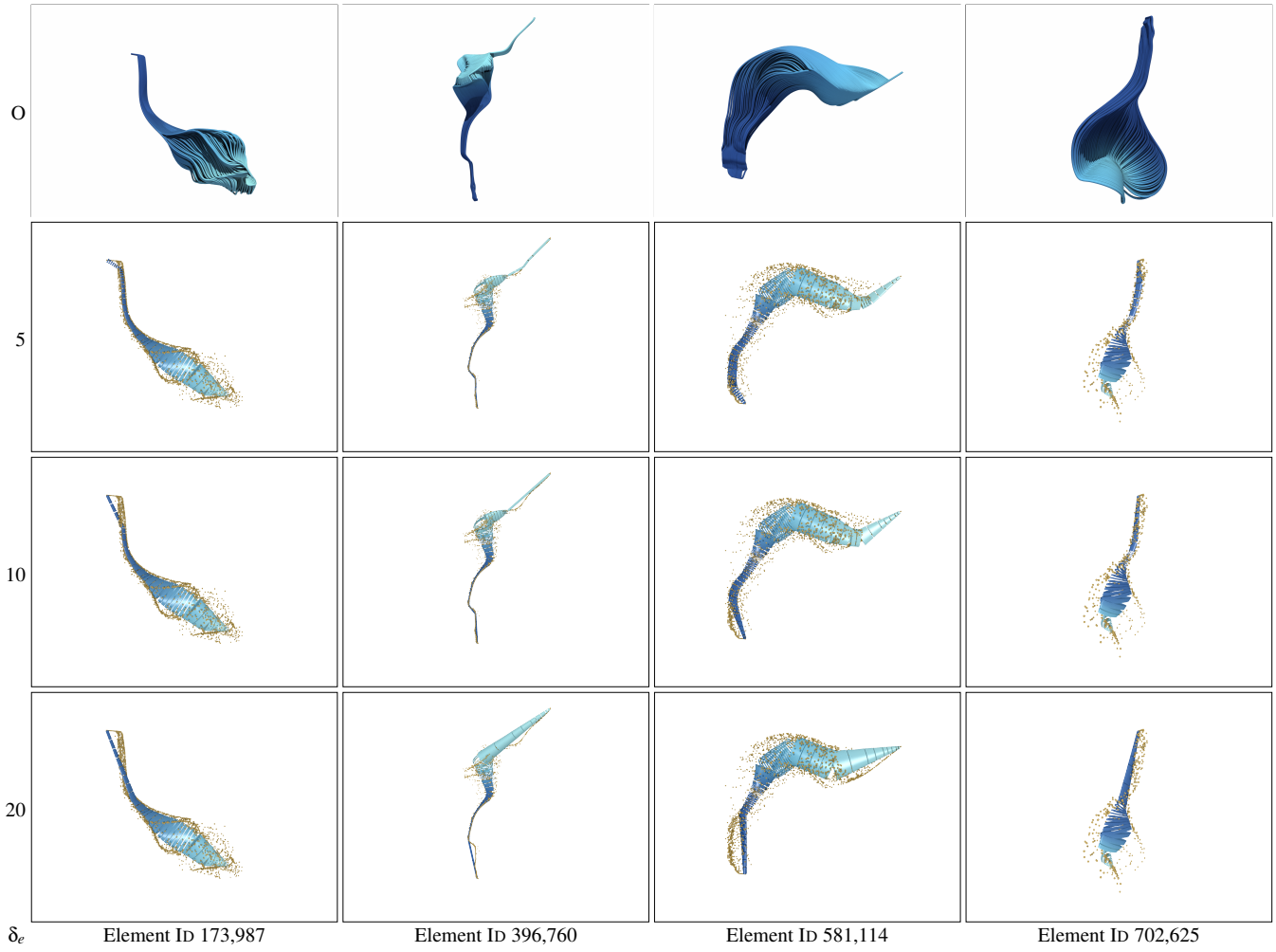

Figure 2: *Overview of the approximation for four selected dissipation elements. The top row shows the original, full trajectory geometry as given in the input data. The other three rows give the approximation for an error threshold of* $\delta_e = 5, 10, 20$*, respectively. For reference, we included samples (yellow) from the input data into the renderings of the approximations.*

- a single socket Intel® Xeon Phi™ 7230 [Knights Landing] processor (KNL) featuring 64 cores at 1.3 GHz, 16 GB of MCDRAM(flat mode) and 96 GB of DDR4 RAM.

The presented algorithm was compiled using

- Embree v2.13 from Intel
- OSPRay v1.1.2 from Intel
- Intel® C++ compiler v17.0.1 with TBB included
- Intel® SPMD Program Compiler v1.9.1

Each element is loaded in its approximated form as presented in Section 3. In order to construct the implicit surface, the set of ellipses is converted into a set of tubelets as presented in Section 4. In order to avoid errors due to interpolating the tubelet's profile, some tubelets are further subdivided on import. Consequently, the tubelets take more memory than the ellipses. Table 2 presents the memory requirements of each element's ellipses and tubelets, respectively. For reference, the memory requirement of each element's raw input trajectories are listed.

For each element, we automatically positioned the camera in such a way that the element's bounding box fits just inside the view frustum. The element was rendered at a 1024×768 resolution using OSPRay's *scivis* renderer, one directional light source casting shadows, one ambient occlusion sample, and no frame accumulation. We allowed the rendering system do a 100-frame warm-up run prior to measurement, and we then recorded the time for a 100-frame full 360° camera orbit around the element. Consequently, the measurements reflect the frame times that will occur during interactive camera movements. In order to compensate for variation due to, e.g., scheduling, we repeated each measurement 10 times. Table 2 presents the average time spent rendering a single frame for each of the elements.

Figure 2 presents the renderings of the approximated elements at three different error thresholds. Here, we used four frame ac-

cumulations in order to reduce the image noise. Consequently, the images resemble the rendered results that are available four frames after a user stopped interactively moving the camera. For reference, we included renderings from the original input data (top row) and samples from the input data (yellow spheres) in Figure 2.

Rendering times for each element in each approximation level are consistently low. Rendering on both nodes, the Broadwell and the KNL, is equally fast. There might be a slight advantage for the KNL. The achieved frame times leave enough headroom to facilitate interactive renderings with larger screen coverage, of more elements simultaneously, and/or using more samples/accumulations in order to reduce noise.

6. Conclusion and Future Work

In this paper, we have presented an integrated approximation and rendering scheme for dissipation elements, a structure definition from contemporary fluid mechanics research. Our approximation algorithm, which makes use of iterative refinement with user-controlled error bounds, allows domain scientists to store large amounts of DE representations for subsequent analysis. Our rendering scheme then enables them to interactively investigate the resulting representations. By using a ray casting approach, our scheme avoids the costly generation and/or storage of explicit, intermediate geometry. Our results show that the proposed scheme reduces the storage requirements by factors in excess of 200× compared to the original, fully resolved trajectory data. Moreover, using OSPRay as the underlying rendering framework, we have demonstrated a rendering of the resulting DE representations at interactive rates.

There are a number of aspects that we would like to address in future work. First, the current iterative refinement scheme only considers the error w.r.t. the centre line approximation. It does not account for sudden changes in the point distribution around the centre line. Hence, the resulting elliptical cross sections might be rotated against each other, leading to an undesirable "candy-wrapper" effect. Moreover, our scheme currently does not cope well with sudden direction changes of the centre line. In such cases, the resulting elliptical tube sections frequently intersect each other. We currently deliberately chose to depict individual tube sections because joining tubes with non-circular cross-sections is non-trivial. This is a very specific point for future improvements, even more so, if the joints should be generated in a way that enables artifact-free semi-transparent visualization.

Finally, our work has been chiefly motivated by the interactive visualization of dissipation elements. However, the underlying rendering core essentially enables ray casting of tubular geometry with an elliptical profile. Hence, we plan to investigate other use cases for this rendering modality.

Appendix

This appendix provides more details on the intersection test in order to help reproduce our findings.

For deriving Equation 5, let $\mathbf{p}_e$ denote a point on an ellipsoid centred at $\mathbf{c}(v)$ in world space. Then,

$$\mathbf{p}_e - \mathbf{c}_e$$

yields the relative position of that point with respect to the ellipsoid's centre $\mathbf{c}_e$, i.e., the point in an ellipsoid-local coordinate space with the ellipsoid centred at the origin. Since the columns of $\mathbf{Q}$ specify the orientation of the ellipsoid's axes in world space,

$$\mathbf{Q}^\top \cdot (\mathbf{p}_e - \mathbf{c}_e)$$

rotates the arbitrarily oriented, origin-centred ellipsoid into an axis-aligned one. Afterwards, the axis-aligned, origin-centred ellipsoid is scaled into a unit sphere via

$$\mathbf{S}^{-1}\mathbf{Q}^\top \cdot (\mathbf{p}_e - \mathbf{c}_e) \quad , \qquad \text{where} \quad \mathbf{S}^{-1} = \begin{bmatrix} r_1^{-1} & 0 & 0 \\ 0 & r_2^{-1} & 0 \\ 0 & 0 & r_3^{-1} \end{bmatrix} \quad ,$$

with $r_i, i = 1,\dots,3$ denoting the ellipsoid's radii. For a unit sphere $r^2 = 1$. Consequently,

$$\begin{aligned} & \left\| \mathbf{S}^{-1}\mathbf{Q}^\top \cdot (\mathbf{p}_e - \mathbf{c}_e) \right\|^2 = 1 \\ \equiv \quad & \left(\mathbf{S}^{-1}\mathbf{Q}^\top \cdot (\mathbf{p}_e - \mathbf{c}_e)\right)^\top \left(\mathbf{S}^{-1}\mathbf{Q}^\top \cdot (\mathbf{p}_e - \mathbf{c}_e)\right) = 1 \\ \equiv \quad & (\mathbf{p}_e - \mathbf{c}_e)^\top \cdot \mathbf{Q}\mathbf{S}^{-1}\mathbf{S}^{-1}\mathbf{Q}^\top \cdot (\mathbf{p}_e - \mathbf{c}_e) = 1 \quad , \end{aligned}$$

and using $\mathbf{\Lambda} := \mathbf{S}^{-1}\mathbf{S}^{-1}$ and $\mathbf{A} := \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$:

$$\begin{aligned} \equiv \quad & (\mathbf{p}_e - \mathbf{c}_e)^\top \cdot \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top \cdot (\mathbf{p}_e - \mathbf{c}_e) = 1 \\ \equiv \quad & (\mathbf{p}_e - \mathbf{c}_e)^\top \cdot \mathbf{A} \cdot (\mathbf{p}_e - \mathbf{c}_e) = 1 \end{aligned}$$

Finally, in order to test for an intersection between a ray and the swept ellipsoid, $\mathbf{p}_e := \mathbf{r}(t)$ and $\mathbf{c}_e := \mathbf{c}(v)$.

In order to create a concise notation for the implementation, we concatenate t-dependent and t-independent terms in the leftmost part of Equation 5:

$$\begin{aligned} \mathbf{r}(t) - \mathbf{c}(v) &= \mathbf{o} + t \cdot \mathbf{d} - \mathbf{c}_\sigma - f \cdot \mathbf{t}_\sigma - t \cdot e \cdot \mathbf{t}_\sigma \\ &= t \cdot (\mathbf{d} + e \cdot \mathbf{t}_\sigma) + \mathbf{o} - \mathbf{c}_\sigma - f \cdot \mathbf{t}_\sigma \\ &= t \cdot \mathbf{g} + \mathbf{h} \quad . \end{aligned}$$

We also concatenate t-dependent and t-independent terms in $\mathbf{A}(t)$ from Equation 6 for conciseness. Therefore, let

$$2\mathbf{D} := \frac{\mathbf{A}_1 - \mathbf{A}_0}{\|\mathbf{c}_\tau - \mathbf{c}_\sigma\|}$$

Then,

$$\begin{aligned} \mathbf{A}(t) &= \mathbf{A}_0 + f \cdot \mathbf{D} + t \cdot e \cdot \mathbf{D} \\ &= t \cdot \mathbf{B} + \mathbf{C} \quad . \end{aligned}$$

In total, this yields for the intersection test

$$\begin{aligned}
& (t\cdot\mathbf{g}+\mathbf{h})^\top\cdot(t\cdot\mathbf{B}+\mathbf{C})\cdot(t\cdot\mathbf{g}+\mathbf{h})=1\\
\equiv\quad & t^3\cdot\mathbf{g}^\top\mathbf{B}\mathbf{g}+t^2\cdot\mathbf{g}^\top\mathbf{B}\mathbf{h}+t^2\cdot\mathbf{g}^\top\mathbf{C}\mathbf{g}+\\
& t\cdot\mathbf{g}^\top\mathbf{C}\mathbf{h}+t^2\cdot\mathbf{g}^\top\mathbf{B}\mathbf{h}+t\cdot\mathbf{h}^\top\mathbf{B}\mathbf{h}+\\
& t\cdot\mathbf{g}^\top\mathbf{C}\mathbf{h}+\mathbf{h}^\top\mathbf{C}\mathbf{h}=1\\
\iff\quad & t^3\cdot\mathbf{g}^\top\mathbf{B}\mathbf{g}+t^2\cdot(2\cdot\mathbf{g}^\top\mathbf{B}\mathbf{h}+\mathbf{g}^\top\mathbf{C}\mathbf{g})+\\
& t\cdot(\mathbf{h}^\top\mathbf{B}\mathbf{h}+2\cdot\mathbf{g}^\top\mathbf{C}\mathbf{h})+\mathbf{h}^\top\mathbf{C}\mathbf{h}-1=0\\
\equiv\quad & a_3\cdot t^3+a_2\cdot t^2+a_1\cdot t+a_0=0\quad,
\end{aligned}$$

a cubic equation being solved using Blinn's numerically stable algorithm [Bli07].

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 720270 (HBP SGA1), from the German Research Foundation (DFG) under grant agreement No KU 1132/10-1, and from the Excellence Initiative of the German federal and state governments.

References

[AO12] ALDUDAK F., OBERLACK M.: Dissipation element analysis in turbulent channel flow. *Journal of Fluid Mechanics 694* (2012), 332–351. 1

[Bli07] BLINN J. F.: How to solve a cubic equation, part 5: Back to numerics. *IEEE Computer Graphics and Applications 27*, 3 (2007), 78–89. 5, 9

[BPL*12] BROWNLEE C., PATCHETT J., LO L.-T., DEMARLE D., MITCHELL C., AHRENS J., HANSEN C.: A Study of Ray Tracing Large-scale Scientific Data in two widely used Parallel Visualization Applications. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV âĂŹ12)* (2012), pp. 51–60. 2

[BSG*11] BERR N., SCHMIDL D., GÖBBERT J. H., LANKES S., AN MEY D., BEMMERL T., BISCHOF C. H.: Trajectory-Search on ScaleMP's vSMP Architecture. In *PARCO* (2011), pp. 227–234. 1, 3

[BSP06] BIGLER J., STEPHENS A., PARKER S. G.: Design for Parallel Interactive Ray Tracing Systems. In *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing* (2006), pp. 187–196. 2

[BWP*10] BREMER P.-T., WEBER G. H., PASCUCCI V., DAY M., BELL J. B.: Analyzing and Tracking Burning Structures in Lean Premixed Hydrogen Flames. *IEEE Transactions on Visualization and Computer Graphics 16*, 2 (2010), 248–260. 2

[CMNR07] CLYNE J., MININNI P., NORTON A., RAST M.: Interactive desktop analysis of high resolution simulations: Application to turbulent plume dynamics and current sheet formation. *New Journal of Physics 9*, 8 (2007). 2

[EHNP03] EDELSBRUNNER H., HARER J., NATARAJAN V., PASCUCCI V.: Morse-smale Complexes for Piecewise Linear 3-Manifolds. In *Proceedings of the Annual Symposium on Computational Geometry* (San Diego, CA, 2003), ACM Press, pp. 361–370. 1

[Fav16] FAVREAU C.: Bluebrain/brayns: Visualizer for large-scale and interactive ray-tracing of neurons. https://github.com/BlueBrain/Brayns, 2016. 2

[GBHP08] GYULASSY A., BREMER P.-T., HAMANN B., PASCUCCI V.: A Practical Approach to Morse-Smale Complex Computation: Scalability and Generality. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1619–1626. 1

[GBP12] GYULASSY A., BREMER P.-T., PASCUCCI V.: Computing Morse-Smale Complexes with Accurate Geometry. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2014–2022. 1

[GSNP13] GAMPERT M., SCHÄFER P., NARAYANASWAMY V., PETERS N.: Gradient trajectory analysis in a jet flow for turbulent combustion modeling. *J. Turbul. 14* (2013), 147–164. 5

[HTP*08] HENTSCHEL B., TEDJO I., PROBST M., WOLTER M., BEHR M., BISCHOF C., KUHLEN T.: Interactive Blood Damage Analysis for Ventricular Assist Devices. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1515–1522. 2

[Kin04] KINDLMANN G.: Superquadric Tensor Glyphs. In *Proceedings of the Sixth Joint Eurographics-IEEE TCVG Conference on Visualization* (2004), Eurographics Association, pp. 147–154. 2

[KW06] KINDLMANN G., WESTIN C.-F.: Diffusion Tensor Visualization with Glyph Packing. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1329–1335. 2

[LAB16] LUCA S., ATTILI A., BISETTI F.: Direct numerical simulation of turbulent lean methane-air bunsen flames with mixture inhomogeneities. In *54th AIAA Aerospace Sciences Meeting (AIAA 2016-0189)* (2016), p. 0189. 5

[LBM*06] LANEY D., BREMER P.-T., MASCARENHAS A., MILLER P., PASCUCCI V.: Understanding the Structure of the Turbulent Mixing Layer in Hydrodynamic Instabilities. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1053–1060. 2

[LGP*15] LI S., GRUCHALLA K., POTTER K., CLYNE J., CHILDS H.: Evaluating the efficacy of wavelet configurations on turbulent-flow data. In *Proceedings of the IEEE Symposium on Large Data Analysis and Visualization* (2015). 2

[NO02] NAKAMARU K., OHNO Y.: Ray tracing for curves primitive. In *Proceedings of Winter School of Computer Graphics (WSCG)* (2002), pp. 311–316. 3

[PBD*10] PARKER S. G., BIGLER J., DIETRICH A., FRIEDRICH H., HOBEROCK J., LUEBKE D., MCALLISTER D., MCGUIRE M., MORLEY K., ROBISON A.: OptiX: A General Purpose Ray Tracing Engine. *ACM Transactions on Graphics (Proceedings ACM SIGGRAPH) 29*, 4 (2010). 2

[PM12] PHARR M., MARK B.: ISPC: A SPMD Compiler for High-Performance CPU Programming. In *Proceedings of Innovative Parallel Computing (inPar)* (2012), pp. 184–196. 2

[PPL*99] PARKER S., PARKER M., LIVNAT Y., SLOAN P.-P., HANSEN C., SHIRLEY P.: Interactive Ray Tracing for Volume Visualization. *IEEE Computer Graphics and Applications 5*, 3 (1999), 238–250. 2

[PSL*98] PARKER S., SHIRLEY P., LIVNAT Y., HANSEN C., SLOAN P.-P.: Interactive ray tracing for isosurface rendering. In *Proceedings of the Conference on Visualization '98* (Los Alamitos, CA, USA, 1998), VIS '98, IEEE Computer Society Press, pp. 233–238. 2

[RJP00] REINDERS F., JACOBSON M. E. D., POST F. H.: Skeleton Graph Generation for Feature Shape Description. In *Proceedings of the Joint Eurographics/IEEE VGTC Symposium on Visualization* (2000), Springer Verlag, pp. 73–82. 2

[SDKS11] SCHÄFER L., DIERKSHEIDE U., KLAAS M., SCHRÖDER W.: Investigation of dissipation elements in a fully developed turbulent channel flow by tomographic particle-image velocimetry. *Physics of Fluids 23*, 3 (2011), 035106. 1

[SGS13] SCHÄFER L., GÖBBERT J. H., SCHRÖDER W.: Dissipation Element Analysis in Experimental and Numerical Shear Flow. *European Journal of Mechanics - B/Fluids 38* (2013), 85–92. 1

[SN12] SHIVASHANKAR N., NATARAJAN V.: Parallel Computation of 3D Morse-Smale Complexes. *Computer Graphics Forum 31*, 3pt1 (2012), 965–974. 1

[TBR*12] TREIB M., BÜRGER K., REICHL F., MENEVEAU C., SZALAY A., WESTERMANN R.: Turbulence visualization at the terascale on desktop pcs. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2169–2177. 2

[vWPSP96] VAN WALSUM T., POST F. H., SILVER D., POST F. J.: Feature Extraction and Iconic Visualization. *IEEE Transactions on Visualization and Computer Graphics 2*, 2 (1996), 111–119. 2

[WBS02] WALD I., BENTHIN C., SLUSALLEK P.: *OpenRT - A Flexible and Scalable Rendering Engine for Interactive 3D Graphics*. Tech. rep., Saarland University, 2002. Available at http://graphics.cs.uni-sb.de/Publications. 2

[WBW*14] WOOP S., BENTHIN C., WALD I., JOHNSON G. S., TABELLION E.: Exploiting Local Orientation Similarity for Efficient Ray Traversal of Hair and Fur. In *High Performance Graphics* (2014), pp. 41–49. 3

[WJA*17] WALD I., JOHNSON G., AMSTUTZ J., BROWNLEE C., KNOLL A., JEFFERS J., GÜNTHER J., NAVRATIL P.: Ospray - a cpu ray tracing framework for scientific visualization. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (Jan 2017), 931–940. 2

[WP06] WANG L., PETERS N.: The Length-Scale Distribution Function of the Distance Between Extremal Points in Passive Scalar Turbulence. *Journal of Fluid Mechanics 554* (2006), 457–475. 1

[WP08] WANG L., PETERS N.: Length-scale distribution functions and conditional means for various fields in turbulence. *Journal of Fluid Mechanics 608* (2008), 113–138. 1, 3

[WP13] WANG L., PETERS N.: A new view of flow topology and conditional statistics in turbulence. *Phil. Trans. of Roy. Soc. 371*, 20120169 (2013). 5

[WSBW01] WALD I., SLUSALLEK P., BENTHIN C., WAGNER M.: Interactive Rendering with Coherent Ray Tracing. *Computer Graphics Forum 20*, 3 (2001), 153–164. (Proceedings of Eurographics 2001). 2

[WWB*14] WALD I., WOOP S., BENTHIN C., JOHNSON G. S., ERNST M.: Embree: A kernel framework for efficient CPU ray tracing. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) 33* (2014). 2