On-The-Fly Tracking of Flame Surfaces for the Visual Analysis of Combustion Processes

T. Oster[1], A. Abdelsamie[1], M. Motejat[1], T. Gerrits[1], C. Rössl[1], D. Thévenin[1], and H. Theisel[1]

[1]University of Magdeburg, Germany

Abstract

The visual analysis of combustion processes is one of the challenges of modern flow visualization. In turbulent combustion research, the behavior of the flame surface contains important information about the interactions between turbulence and chemistry. The extraction and tracking of this surface is crucial for understanding combustion processes. This is impossible to realize as a post-process because of the size of the involved datasets, which are too large to be stored on disk. We present an on-the-fly method for tracking the flame surface directly during simulation and computing the local tangential surface deformation for arbitrary time intervals. In a massively parallel simulation, the data is distributed over many processes and only a single time step is in memory at any time. To satisfy the demands on parallelism and accuracy posed by this situation, we track the surface with independent micro-patches and adapt their distribution as needed to maintain numerical stability. With our method, we enable combustion researchers to observe the detailed movement and deformation of the flame surface over extended periods of time and thus gain novel insights into the mechanisms of turbulence-chemistry interactions. We validate our method on analytic ground truth data and show its applicability on two real-world simulation.

CCS Concepts

•***Human-centered computing*** $\rightarrow$ ***Scientific visualization;*** •***Computing methodologies*** $\rightarrow$ *Massively parallel algorithms;*

1. Introduction

A central topic of today's combustion research are the interactions between turbulent flow and chemical reactions in flames. In this context, the flame surface, i.e., the thin region where reaction happens, is of central importance. Temperature, concentrations of chemical species and other variables on the flame surface show the state of the reaction. The transport and deformation of the flame surface over time show the joint influence of chemistry and turbulence on the combustion process.

To study the detailed mechanisms of turbulent combustion, direct numerical simulations (DNS) are used. These simulations directly compute the fundamental physical and chemical equations on a high-resolution grid without any high-level modeling assumptions. As a consequence, all physical and chemical quantities can be analyzed in detail. However, the computational costs are very high and the data sets produced can easily reach into the tera- or petabyte range. This makes them practically impossible to store completely or transfer over a network connection. For this reason, only single snapshots of the simulation are commonly stored on disk and analyzed in post-processing. This approach only allows investigating the state of the flame surface at single points in time. The temporal distance between subsequent snapshots is generally too large to interpolate between them in any meaningful way or make any statement about the correspondence of surface points.

In this work, we introduce an algorithm for tracking the flame surface on-the-fly during the simulation, while the necessary data is still in memory. Our goal is to capture the shape of the surface, the history of individual surface points, and the local tangential deformation of the surface over extended periods of time. This allows us to expand studies of the history of single points on the surface over extended time periods, such as described by Sripakagorn et al. [SMKP04], to the complete flame surface. Additionally, we extend the notion of instantaneous surface stretch (such as described by Poinsot and Veynante in [PV12]) to the tangential deformation of the surface over arbitrary time intervals.

Our algorithm must overcome the following challenges:

1. The massively parallel simulation, distributes its domain across a large number of processes. The surface tracking must be similarly parallelizeable.
2. Performing analysis on-the-fly during a simulation means that at any point in time, only data for the current time step is available in memory and there is no way of going back in time to retrieve previous information.
3. The surface is expected to undergo significant deformation over time, making an adaptive refinement and coarsening necessary.
4. The tangential deformation must be reconstructed for areas of the surface that have been refined and/or coarsened multiple times over an arbitrary time interval.

Point 1. precludes the use of a mesh or space partition data structure with neighborhood information, as the global nature of operations in such a structure is not well suited for a massively parallel algorithm. Instead, we represent the surface as a number of independent micro-patches consisting of a central point and four ghost particles measuring the local surface deformation. This is described in Section 4.1. Points 2. and 3. mean that we need to refine the micro-patches adaptively before they are significantly distorted. We therefore introduce a way of monitoring the distortion of a patch and split it before the distortion becomes too large in Section 4.2. This method of tracking and refining a surface without explicit neighbor information is a major contribution of this work. Based on the behavior of the micro-patches over time, we introduce the computation of the tangential deformation gradient (point 4.) for arbitrary time intervals in Section 4.3.

2. Background and Related Work

The flame surface is the location of a flame where chemical reactions take place. In premixed flames, where fuel and oxidizer are mixed before ignition, it is the interface between burnt and unburnt gases. In non-premixed combustion, it is the burning part of the interface between fuel and oxidizer. The flame surface is defined as an isosurface of a scalar variable such as the temperature or the mixture fraction of fuel and oxidizer. Over the course of the simulation, the flame surface is influenced by multiple factors. Chemical reactions move the interface between gases by transforming them, turbulent flow transports the gases, moving and deforming the flame surface, and molecular and thermal diffusion have a smoothing effect on the surface shape. An example of the effects of tangential stretching on the burning behavior of the flame is given by Renard et al. in [RRTC99]. For a more detailed introduction to turbulent combustion, see “Theoretical and Numerical Combustion” by Poinsot and Veynante [PV12]. A modern direct numerical simulation code is presented by Abdelsamie et al. in [AFO*16].

The evolution of simulation variables on the path of single points on the flame surface has been used in multiple works in combustion literature [SMKP04, SDGH17]. In these works only a small number of points on the surface are tracked and they do not consider the relative tangential movement of points. Our approach allows these kinds of studies on a much larger scale, providing data for the whole surface, which enables statistical evaluation and visual analysis. Stretching of a flow restricted to a surface has been investigated similarly to our approach by Garth et al. [GWT*08].

Tracking different kinds of features has been the subject of numerous works in the field of flow visualization. A lot of research deals with tracking volumetric features over time [SX97, SYM14, CMN13, MGYC09, DHS*12, MM09], while some methods focus on point- [GTS04, TS03] or line-type features [BWP*10]. Most of these methods are designed for datasets that fit into the main memory of a consumer-grade computer, although some are explicitly designed to work in distributed-memory settings [WYM13].

Surface extraction and tracking is another large field of research. In the context of this work, particle-based isosurface extraction methods, like the one proposed by Crossno et al. [CA97], as well as methods for tracking evolving surfaces over time [KGJ09, BFTW09, BOJH15] are relevant. Of particular interest in the context of this work is the work by Camp et al. [CCG*12], wich deals with stream surface integration in a distributed-memory environment.

Visualization methods designed for an on-the-fly setting in large scale parallel simulations include the rendering of volume, surface and point data [AMCH07, YWG*10]. These provide an overview of the state of a simulation while it is running and allow a visual analysis of the data for all time steps without saving them to disk. Agranovsky et al. [ACG*14] proposed storing flow fields computed during a simulation as a collection of path lines. This Lagrangian form allows a compressed storage of the flow without relying on snapshots with large temporal gaps.

To the best of our knowledge, there is no existing approach that solves the problem of tracking a time surface in a large distributed-memory simulation while maintaining temporal correspondence between surface points.

3. Mathematical Basis

In the following two sections, we give a brief introduction into the mathematical basis of the problems we want to solve. First, we derive the equation for the movement of the flame surface over time. Then, we introduce the tangential deformation gradient, which describes the distortion an infinitesimally small section of the surface experiences in a certain time interval.

3.1. Tracking the Flame Surface

We view the flame surface as an implicit surface $s(\mathbf{x},t) = \Gamma$. This scalar function is transported by the fluid velocity $\mathbf{v}(\mathbf{x},t)$ and influenced by diffusion and chemical reaction processes. A point $\mathbf{x}$ with velocity $\mathbf{u}$ tracking the surface has a zero Lagrangian derivative, i.e., the value of s at the point does not change over time. Therefore

$$\frac{\mathrm{D}s}{\mathrm{D}t} = \frac{\partial s}{\partial t} + \nabla s \cdot \mathbf{u} = 0\,.$$

This constrains the component of $\mathbf{u}$ that is normal to the surface. In tangential direction, we want $\mathbf{u}$ to adhere to the fluid velocity, i.e.,

$$\|\mathbf{u} - \mathbf{v}\|^2 \to \min\,.$$

Combining both constraints using Lagrange multipliers, the solution for the velocity of a point on the implicit surface is given by

$$\mathbf{u} = \mathbf{v} - \frac{\frac{\partial s}{\partial t} + \nabla s \cdot \mathbf{v}}{\|\nabla s\|^2} \nabla s\,.$$

This equation is expressed as $\mathbf{u} = \mathbf{v} + s_d \mathbf{n}$ in combustion literature [PV12]. Here, s_d is the speed of flame propagation normal to the surface and relative to the fluid velocity and $\mathbf{n} = -\nabla s / \|\nabla s\|$ is the unit surface normal.

3.2. Tangential Deformation of an Implicit Surface in a Flow

In the previous section, we showed how a point on the surface moves over time. We now derive the tangential deformation gradient, which encodes the relative movement of surface points in an infinitesimal neighborhood.

We consider the starting position of a point on the surface at

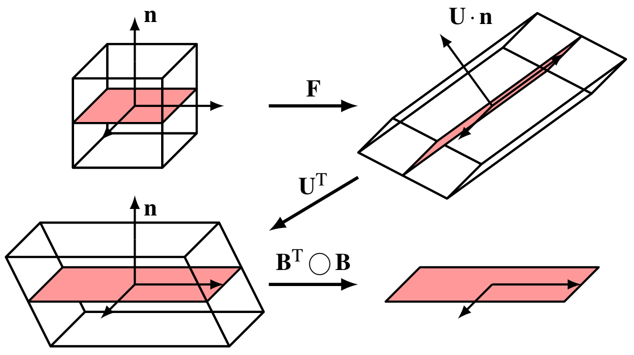


Figure 1: Extracting the tangential component of the deformation gradient. A local neighborhood is transformed by the deformation gradient **F**. In the process, the local coordinate system is rotated by **U**, which must be reverted before isolating the tangential component of **F** by multiplying with the tangential basis **B** from both sides.

some time t_0. W.l.o.g. we assume that this point is at the origin $\mathbf{0}$. Let $\chi(\mathbf{x},t)$ be the mapping function that maps a point $\mathbf{x}$ to its position at time $t > t_0$ after moving with the surface. W.l.o.g. we assume that $\chi(\mathbf{0},t) = \mathbf{0}$. The spatial deformation gradient $\mathbf{F} = \nabla\chi$, which encodes the behavior of a point $\mathbf{x}$ in an infinitesimally small neighborhood around $\mathbf{0}$, can then be expressed as

$$\mathbf{F} \cdot \mathbf{x} = \chi(\mathbf{x},t) \quad \text{for } \mathbf{x} \to \mathbf{0}.$$

We are only interested in the tangential part of this deformation, i.e., the behavior in a tangential coordinate system that moves and rotates with the surface. In order to isolate the tangential component, we first need to eliminate this rotation. A (right) polar decomposition $\mathbf{F} = \mathbf{U}\mathbf{P}$ separates the rotational part $\mathbf{U}$ from the rest of the deformation. The tangential component is now obtained by projecting $\mathbf{P} = \mathbf{U}^{\mathrm{T}}\mathbf{F}$ into the local tangent space at time t_0. Let $\mathbf{B}$ be a matrix with orthonormal basis vectors of this tangent space as its columns. Then the tangential deformation gradient $\hat{\mathbf{F}}$ is

$$\hat{\mathbf{F}} = \mathbf{B}^{\mathrm{T}}\mathbf{U}^{\mathrm{T}}\mathbf{F}\mathbf{B}. \qquad (1)$$

Note that $\mathbf{B} \in \mathbb{R}^{3\times 2}$; and thus $\hat{\mathbf{F}} \in \mathbb{R}^{2\times 2}$. See Figure 1 for an intuitive interpretation of this transformation. Thus, $\hat{\mathbf{F}}$ encodes the behavior of a point $\hat{\mathbf{x}}$ in an infinitesimally small neighborhood around $\hat{\mathbf{0}}$ in tangent space:

$$\hat{\mathbf{F}} \cdot \hat{\mathbf{x}} = \mathbf{B}^{\mathrm{T}}\chi(\mathbf{B}\hat{\mathbf{x}},t) \quad \text{for } \mathbf{x} \to \mathbf{0}.$$

4. Discretization

Our objective is an algorithm for tracking the flame surface on-the-fly during a simulation run. It must produce the paths of single points on the surface over time, as well as the tangential deformation of the surface for arbitrary time intervals. Additionally, the algorithm needs to be highly parallelizeable in order to work well in the environment of a massively parallel simulation. A naive approach might be performing an isosurface extraction in each time step of the simulation. However, subsequent isosurfaces do not contain information about surface point correspondence between time steps.

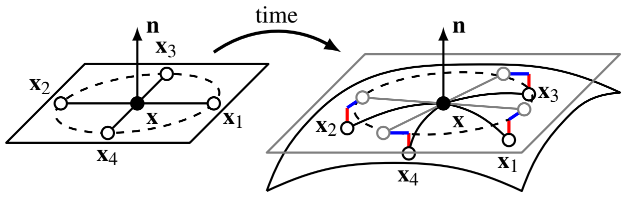


Figure 2: Ghost particles (○) are initialized in an orthogonal configuration around the central point (●) in the tangent plane. While integrating the points over time, they deviate from a planar, linear behavior. The difference between the real positions and the nearest linear configuration (○) in the tangent plane is measured by the normal error $e_{\perp}$ (—) and the tangential error $e_{\shortparallel}$ (—).

This correspondence is necessary if we want to provide point paths and compute the surface deformation. Alternatively, one could advect the vertices of an explicit surface mesh, adaptively remeshing it as the surface deforms over time. Due to the global nature of such remeshing operations, this is infeasible in a massively parallel environment, where irregular communication between neighboring processors is known to be a major source of bottlenecks.

We therefore choose to represent the surface as a cloud of micro-patches consisting of a central point and four *ghost particles* (see Figure 2, left). The ghost particles sample the shape and deformation of the micro-patch in the local neighborhood around the central point. Each group of points is tracked completely independently. We monitor the deviation of the ghost particles from the tangent plane at the central point and the deviation of their relative movement from linear behavior. Once this deviation exceeds a user-defined threshold, we split the patch into three independent new patches to ensure sufficient sampling. To avoid oversampling, we merge patches when they become too small, e.g., because the surface flattens over time or tangential movement bunches up many patches in a small area. The tangential surface deformation is then reconstructed from the relative behavior of the points in a group between split/merge events.

4.1. Micro-Patches for Surface Tracking

A micro-patch is represented by a group of five points: A center and four *ghost particles*. On initialization, the points are arranged in an orthogonal cross shape in the tangent plane defined by the surface normal $\mathbf{n}$ at the central point $\mathbf{x}$ (see Figure 2). In this way, they sample the shape and deformation in all directions as they follow the surface over time. We describe the configuration (e.g., the state excluding the absolute position) of a micro-patch at time t as a matrix $\mathbf{C}(t) \in \mathbb{R}^{3\times 5}$ consisting of the surface normal $\mathbf{n}(t)$ at the central point as well as the relative positions of the ghost particles $\mathbf{x}_i(t),\ i \in \{1,\dots,4\}$.

$$\mathbf{C}(t) = \begin{pmatrix} \mathbf{n}(t) & \mathbf{x}_1(t) - \mathbf{x}(t) & \cdots & \mathbf{x}_4(t) - \mathbf{x}(t) \end{pmatrix}.$$

In the following, we omit the dependence of $\mathbf{C}$, $\mathbf{n}$, and $\mathbf{x}_i$ on the time t and we assume that $\mathbf{x} = \mathbf{0}$ wherever the meaning is clear from context. For the initial configuration at some time t_0, $\mathbf{C}$ can be exactly represented by a unit base configuration $\mathbf{C}_0$ being transformed by a linear transformation $\mathbf{T}$, i.e.,

$$\mathbf{C}(t_0) = \mathbf{T}\,\mathbf{C}_0, \quad \text{with} \quad \mathbf{C}_0 = \begin{pmatrix} \mathbf{e}_3 & \mathbf{e}_1 & -\mathbf{e}_1 & \mathbf{e}_2 & -\mathbf{e}_2 \end{pmatrix},$$

where $\mathbf{e}_i$ are the unit vectors in the i-th coordinate direction. As the points track the surface over time, their relative position will change. Their relation to $\mathbf{C}_0$ might no longer be linear. Now, $\mathbf{T}$ is the linear transformation that best approximates the mapping between $\mathbf{C}_0$ and $\mathbf{C}$, i.e., the solution to the least squares problem

$$e^2 = \left\| \mathbf{C}_0^{\mathrm{T}} \mathbf{T}^{\mathrm{T}} - \mathbf{C}^{\mathrm{T}} \right\|_F^2 \to \min .$$

Here, $\|\cdot\|_F$ denotes the Frobenius norm. The residual error e measures the deviation of the real mapping between $\mathbf{C}_0$ and $\mathbf{C}$ from linear behavior. However, its scale is dependent on the size of the micro-patch, i.e., the magnitude of the last four columns of $\mathbf{C}$, and it weights those columns differently from the normal in the first column. To get a more meaningful error, we normalize these columns by their average magnitude and obtain a modified transformation $\mathbf{T}_n$ and error e_n by

$$e_n^2 = \left\| \mathbf{C}_0^{\mathrm{T}} \mathbf{T}_n^{\mathrm{T}} - \mathbf{C}_n^{\mathrm{T}} \right\|_F^2 \to \min , \tag{2}$$

$$\text{with } \mathbf{C}_n = \begin{pmatrix} \mathbf{n} & \mathbf{x}_1/c & \cdots & \mathbf{x}_4/c \end{pmatrix} \text{ and } c = \frac{1}{4}\sum_i^4 \|\mathbf{x}_i\| .$$

We separate this error into normal and tangential components

$$e_\perp = \left\| \mathbf{n}\mathbf{n}^{\mathrm{T}} (\mathbf{T}_n \mathbf{C}_0 - \mathbf{C}_n) \right\|_F$$
$$e_\shortparallel = \left\| (\mathbf{I} - \mathbf{n}\mathbf{n}^{\mathrm{T}})(\mathbf{T}_n \mathbf{C}_0 - \mathbf{C}_n) \right\|_F .$$

$e_\perp$ is a measure for the deviation of the ghost particles $\mathbf{x}_i$ from the tangent plane. As such, it measures the local surface curvature in relation to the patch size. $e_\shortparallel$ measures how much the tangential deformation of the ghost particles deviates from linear behavior. As the micro-patch changes over time, we use these errors to decide when to split or merge patches to ensure sufficient sampling of the surface geometry and tangential deformation.

4.2. Splitting and Merging Surface Patches

We split a micro-patch into three new patches when one of the following occurs:

- The error $e_\perp$ exceeds a threshold $r_\perp$
- The error $e_\shortparallel$ exceeds a threshold $r_\shortparallel$
- The major axis exceeds a threshold r_{size}

The first two are to ensure a sufficient sampling, the third is to ensure a maximum patch size even on flat parts of the surface. In a parallel distributed simulation, the size of a patch is limited by the size of the block of the simulation domain it is contained in.

When splitting a micro-patch, we need to decide the positions of the points forming the new patches. In this context, it helps to think of the micro-patch as an ellipse being defined by the positions of the ghost particles around the central point. On initialization, the ghost particles are always placed along the principal axes of the ellipse. When the patch is transformed over time, the principal axes will generally not stay aligned with the ghost particles. The major and minor axes of the current configuration can be reconstructed from the tangential component $\hat{\mathbf{T}}$ of $\mathbf{T}$, which is obtained similar to (1):

$$\hat{\mathbf{T}} = \begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_2 \end{pmatrix}^{\mathrm{T}} \mathbf{U}^{\mathrm{T}} \mathbf{T} \begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_2 \end{pmatrix} ,$$

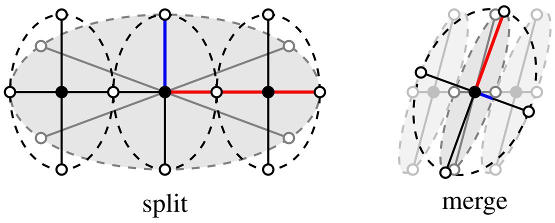


Figure 3: Splitting and merging micro-patches. Left: An old micro-patch (◌) is split up along its major axis (—) into three new identical patches (◌). Right: An old micro-patch (◌) is extended to three times its size along its minor axis (—). Neighboring patches (◌) that were split off from the central one at some earlier time are deleted instead.

where $\mathbf{U}^{\mathrm{T}}$ is the rotation obtained from the polar decomposition $\mathbf{T} = \mathbf{UP}$. The directions and extents of the principal axes of the micro-patch are encoded in the singular value decomposition

$$\hat{\mathbf{T}} = \hat{\mathbf{V}} \hat{\Sigma} \hat{\mathbf{W}}^{\mathrm{T}} .$$

The singular values $\sigma_{1,2}$ on the diagonal of $\hat{\Sigma}$ are the extents of the ellipse, while the rows of $\hat{\mathbf{V}}$ are the directions of the major and minor axis in tangent space. The directions in 3D space are the columns of $\mathbf{U} \begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_2 \end{pmatrix} \hat{\mathbf{V}}^{\mathrm{T}}$.

When splitting the micro-patch, we want the new patches to cover the whole area captured by the old patch without overlapping, to prevent over- or undersampling as patches are split multiple times. We also want to leave the central point in place, so we can track its path for as long as possible. Consequently, we split the patch along its major axis into three identical new ones (see Figure 3). The points of the new patches are again arranged in an orthogonal cross shape with the axes parallel to the axes of the old patch. The length of the new first axis is exactly $1/3$ of the length of the old major axis. The new second axis is as long as the old minor axis. The new points are then projected onto the surface along the normal at the central point. From this point on, they are treated idependently again.

The new central patch continues tracking the behavior of the neighborhood around the same point as the old patch. We therefore consider it to be the same entity, but with its ghost particles reset to a more numerically stable configuration. In contrast, we consider the new outer patches to be entirely new entities. Consequently, if we say that a patch has been split or merged multiple times, we mean that it has been the central patch in a number of these operations.

We consider a micro-patch for merging when

- both errors $e_\perp$ and $e_\shortparallel$ decrease below lower thresholds $l_\perp$ and $l_\shortparallel$, respectively,
- or the minor axis decreases below a threshold l_{size}

Because the micro-patches are completely independent, we cannot explicitly merge multiple patches. We therefore give every patch a counter, which is set to 0 when it is first initialized and incremented each time it is split. When the conditions for a merge are met, the behavior of a patch depends on its counter. If the counter is 0, the patch is simply deleted. Otherwise, the counter is decremented and the patch is extended by a factor of 3 along its minor axis. Assuming the behavior of neighboring patches is similar, this effectively means that the new extended patch now covers the area of the neighboring patches, which were deleted. This assumption does not generally hold for all parts of the surface. We therefore typically choose $l_\perp$, $l_{\shortparallel}$, and l_{size} to be very small. In this case, points will only be deleted in areas of extreme tangential compression, where removing very small patches will only result in small errors, and in areas of very simple deformation, where the assumption is unlikely to be violated.

The central point of the extended patch stays in place, while the ghost particles are again placed along the principal axes of the old patch, but with the minor axis extended. In this way, we ensure that the surface is not over-sampled where it is not necessary and that no infeasibly small patches are tracked, e.g., in areas of attracting tangential behavior. Patches that were originally initialized at the start of the simulation are never deleted, in order to prevent holes from forming. Since the surface at the start of a simulation is generally very simple and can be represented by a relatively small number of patches, this is not an issue in practice.

4.3. Reconstructing Tangential Surface Deformation

Let $\hat{\mathbf{F}}_{t_s}^{t_e}$ be the tangential deformation gradient for a time interval $[t_s, t_e]$ at the central point $\mathbf{x}(t_e)$. To reconstruct it we need the corresponding deformation gradient $\mathbf{F}_{t_s}^{t_e}$, which describes the relative change in position of points in a small neighborhood between times t_s and t_e. The behavior of the ghost particles of a micro-patch over time contains exactly this information. However, because all ghost particles are located on the surface, they do not contain information about the behavior of $\mathbf{F}$ in normal direction. Fortunately, we only need the change in orientation of the surface normal to reconstruct $\hat{\mathbf{F}}$, as any other information is discarded when projecting into the tangent plane. It is therefore sufficient to determine a proxy transformation $\mathbf{E}_{t_s}^{t_e}$, which maps the micro-patch configuration $\mathbf{C}(t_s)$ to $\mathbf{C}(t_e)$, and reconstruct $\hat{\mathbf{F}}$ via

$$\hat{\mathbf{F}}_{t_s}^{t_e} = \mathbf{B}^{\mathrm{T}} \mathbf{U}^{\mathrm{T}} \mathbf{E}_{t_s}^{t_e} \mathbf{B},$$

with $\mathbf{B}$ the basis vectors of the tangent plane at t_s and $\mathbf{U}$ the rotation matrix from the polar decomposition of $\mathbf{E}$.

Let us first assume the micro-patch was not split or merged in the time interval, i.e., its ghost particles were not reset. Because the real mapping between $\mathbf{C}(t_s)$ and $\mathbf{C}(t_e)$ will generally not be linear, $\mathbf{E}_{t_s}^{t_e}$ is the solution to a least squares problem very similar to (2). To make the solution independent of the scale of the micro-patch, we scale the last four columns of both $\mathbf{C}(t_s)$ and $\mathbf{C}(t_e)$ by the average norm of these columns in $\mathbf{C}(t_e)$. The system for reconstructing $\mathbf{E}_{t_s}^{t_e}$ is then

$$\left\| \mathbf{C}_n^{\mathrm{T}}(t_s) \mathbf{E}_{t_s}^{t_e\,\mathrm{T}} - \mathbf{C}_n^{\mathrm{T}}(t_e) \right\|_F^2 \to \min, \tag{3}$$

with $\mathbf{C}_n(t_s)$ and $\mathbf{C}_n(t_e)$ being the configurations with their last columns scaled.

If the ghost particles were reset one or more times during the time interval, $\mathbf{E}_{t_s}^{t_e}$ is the concatenation of multiple transformations:

$$\mathbf{E}_{t_s}^{t_e} = \mathbf{E}_{t_n}^{t_e} \cdot \mathbf{E}_{t_{n-1}}^{t_n} \cdot \cdots \cdot \mathbf{E}_{t_1}^{t_2} \cdot \mathbf{E}_{t_s}^{t_1},$$

where t_i are the discrete times between t_s and t_e when the ghost particles were reset in the course of a split or merge operation.

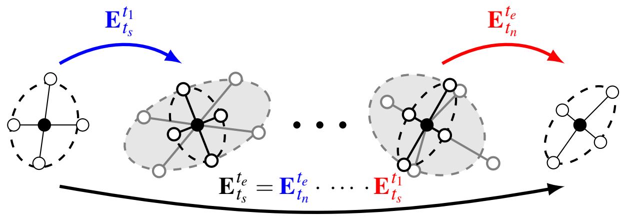

$\mathbf{E}_{t_s}^{t_1}$
$\mathbf{E}_{t_n}^{t_e}$
$\mathbf{E}_{t_s}^{t_e} = \mathbf{E}_{t_n}^{t_e} \cdot \cdots \cdot \mathbf{E}_{t_s}^{t_1}$

Figure 4: Reconstructing the deformation for an arbitrary time interval by concatenating the deformations between split and merge events of the patch.

Each sub-interval transformation is reconstructed from the new configuration of the micro-patch just after a split/merge and the old configuration just before the next one (see Figure 4).

If we want to compute the tangential deformation a micro-patch at time t_e has experienced since the start time t_s, it is possible that this patch has not existed for the complete time interval, i.e., it was created at some time $t_k > t_s$ during a split operation. In this case, we estimate the transformation between t_s and t_k from its parent patch by weighting the contributions of each ghost particle differently depending on their distance from the center of the new patch. The details of this algorithm are presented in Appendix A. With this, we can reconstruct the tangential deformation gradient of any micro-patch for an arbitrary time interval.

4.4. Initialization

At the start of the simulation, the initial surface has to be seeded with patches. Most simulations start with a very simple surface, such as a plane or sphere, for which this operation is trivial. If the starting surface is more complex, any existing isosurface meshing algorithm that produces near-equilateral triangles can be used, with patches initialized to cover the area of the 1-ring of each vertex. When initializing the micro-patches, care has to be taken not to leave any holes, which might grow larger over time if the surface expands. We ensure this by overlapping the initial patches, which will increase the number of necessary patches by a constant factor, but is easy to implement. More elaborate approaches which avoid covering the surface with more than one layer of patches are conceivable.

5. Implementation

Our algorithm is implemented as an extension to the DINO direct numerical simulation code [AFO*16]. The simulation domain is a rectangular box which is distributed to multiple processors by a block decomposition along two of its three axes. This unusual decomposition is required because of the pressure solver, which operates in Fourier space and needs the complete domain in memory in one dimension. The communication between the processors uses MPI [MPI15].

Each tracked points gets a unique ID at its creation. In each process, all points located in its domain block, as well as all points in a halo region around the block are kept in memory. This halo region is large enough to contain all ghost particles of any patch whose central point is inside the processor's block.

After each simulation step, the fluid velocity $\mathbf{v}$ and derivatives of the scalar variable s defining the flame surface are interpolated from the simulation grid at all surface points to compute their velocity $\mathbf{u}$. Because each simulation block only holds data for the grid cells inside its boundaries, the values for the points in the halo region have to be obtained by communicating with all neighbor processes. The points are then advanced by one simulation time step. For additional stability, we then perform a few steps of a Newton scheme to move them back onto the isosurface we are tracking. As the surface points move during the simulation, they will often migrate across processor boundaries. This is automatically handled by our implementation as part of the information exchange with neighboring processors when updating the points in the halo region.

Since our algorithm is running in lockstep with the simulation, only data for the current time step is available in memory at any time and the time step is controlled by the simulation. This means that without generating additional memory overhead, only first order schemes can be used to integrate the surface velocity $\mathbf{u}$. In our implementation, we therefore use an Euler scheme to advance the positions of the points between simulation steps. The simulation time step in DNS is generally dominated by the chemistry time scale, which is much smaller than the fluid velocity time scale. Since the surface velocity typically follows the fluid velocity closely, and an Euler scheme is adequate to track the flame surface almost everywhere, as points move only a fraction of the size of a grid cell in each time step. Exceptions to this rule only occur at very sharp creases in the surface, which can occur when two parts of the surface fold into each other, possibly leading to changes in topology. The high surface velocities occuring here can lead to points migrating far away from the surface in a single time step. Since the surface at these locations is contracting rapidly anyway, we simply delete any micro-patches that end up further away from the surface than one half of their previous advection step, provided the distance is at least $1/5$ the size of a grid cell. In our tests, this strategy did not impact the accuracy of the surface tracking in any significant way.

Depending on the values of the errors $e_\perp$ and $e_{\shortparallel}$, the micro-patches are now split or merged and any patches that partially crossed a non-periodic outflow boundary of the simulation domain are removed. At this point, control is given back to the simulation, which performs the next iteration.

The deformation gradients can either be computed on-the-fly or as a post-processing step. In the first case, the time intervals for the deformation need to be specified before the simulation starts. Each patch then remembers its configuration $\mathbf{C}$ at the time it was last reset and the transformations $\mathbf{E}$ from the start of each time interval up until this last reset time. In the second case, the point positions and normals of all micro-patches are stored to disk for each time step. The deformation gradients can then be reconstructed for arbitrary time intervals, but at the cost of increased hard disk storage demand.

6. Results

We tested our algorithm on an analytic test function that is designed to resemble the behavior of a vortex, and on two real-world simulations. The simulations were carried out on Phase 1 of the SuperMUC Petascale System of the Leibnitz Supercomputing Centre in Garching, Germany. Each node of the system has two 8-core Intel Xeon E5 processors with a clock frequency of 2.7 GHz and 32 GB of shared memory. The nodes are connected via Infiniband FDR10.

In the following sections, we evaluate the accuracy of our method on the analytic test function, and show our results for the two simulation cases.

6.1. Analytical Test Function

To evaluate the accuracy of the tangential deformation gradient obtained by our method, we designed an analytic test function. It imitates the behavior of a vortex in the shear layer between two gas streams. The test function s is defined as

$$s(x,y,z,t) = \begin{cases} z\cos A - (x-\frac{1}{2})\sin A, & \text{for } \sqrt{(x-\frac{1}{2})^2 + z^2} < \dfrac{1}{2} \\ z, & \text{else} \end{cases},$$

$$A(x,y,z,t) = 2\pi t \sin\left(\frac{\pi}{2}\left(1+\sqrt{4x^2-4x+4z^2+1}\right)\right)^2 \sin(\pi y)^2 .$$

The isosurface $s = 0$ coincides with the xy plane at $t = 0$. As t increases, the surface is curled up around the center at $(x,\ y,\ z) = (1/2,\ 1/2,\ 0)$. The underlying velocity field of a vortex can not be easily expressed as an analytic function. We therefore assume a fluid velocity of $\mathbf{v} = \mathbf{0}$ for our tests.

We observe the function in the domain $x \in [0,\ 1]$, $y \in [0,1]$, $z \in [-1/2,\ 1/2]$, $t \in [0,\ 2]$ resolved in a $72 \times 72 \times 72$ regular grid and a time step of $\Delta t = 5 \times 10^{-3}$. This means that in the investigated time span the vortex makes exactly two full turns, resolved in 4000 time steps. This closely resembles the lifetime of a vortex and temporal resolution observed in a real simulation setting.

The velocity $\mathbf{u}$ of the surface $s = 0$ can be expressed as an analytic function. This enables us to obtain highly accurate ground truth data for the tangential deformation gradient $\hat{\mathbf{F}}$. The largest singular value of $\hat{\mathbf{F}}$ signifies the largest stretching experienced by the local neighborhood of a point on the surface, which we call the stretching coefficient c with

$$c = \sigma_{\max}(\hat{\mathbf{F}}) = \sqrt{\lambda_{\max}(\hat{\mathbf{F}}^{\mathrm{T}}\hat{\mathbf{F}})} .$$

This is similar to the measure used when computing the Finite-Time Lyapunov Exponent (FTLE) for measuring separation in flow fields [Hal02]. In all tests, we use a normal error threshold of $r_\perp = 0.5$, which is rather coarse. By doing this, we limit its influence on the accuracy of the results for the tangential deformation. We show the distribution of differences to the ground truth solution for different tangential error thresholds $r_{\shortparallel}$ in Figure 6. For the sake of brevity, we only show the results for the complete interval $t = [0, 2]$, which will naturally show the largest errors. As shown in Figures 5 and 6, the accuracy of our method is strongly dependent on $r_{\shortparallel}$. For large values of $r_{\shortparallel}$, the resulting deformation does not only contain unreasonably large errors, but the surface also exhibits some holes. Splitting too late means that there is potentially a lot of non-uniform stretching across the patch that is not accounted for by the resulting child patches. This shows that observing only the normal error $e_\perp$ is not sufficient to guarantee a good sampling of the surface, even if one is not interested in an accurate result for the tangential deformation

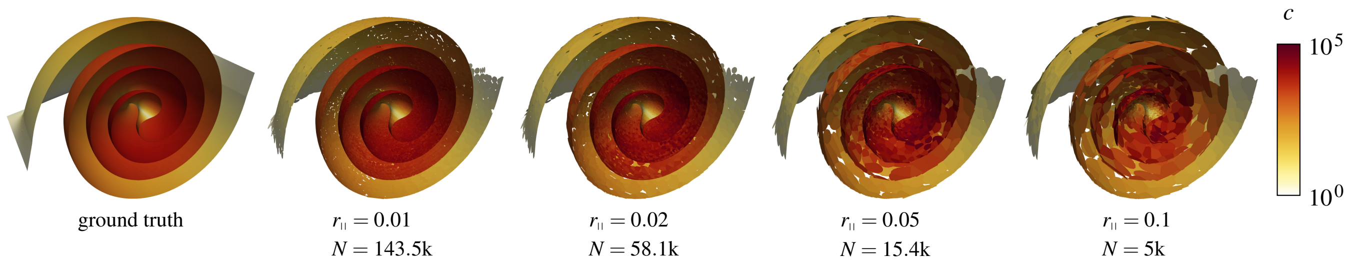

c
10^5
10^0
ground truth
$r_{\shortparallel} = 0.01$
$N = 143.5$k
$r_{\shortparallel} = 0.02$
$N = 58.1$k
$r_{\shortparallel} = 0.05$
$N = 15.4$k
$r_{\shortparallel} = 0.1$
$N = 5$k

Figure 5: The number of patches N and the stretch coefficient c of the tangential deformation gradient $\hat{\mathbf{F}}_0^2$ for the analytic test function, using $r_\perp = 0.5$ and varying values for $r_{\shortparallel}$. Each micro-patch is represented by an ellipse scaled and aligned according to its principal axes and orthgonal to its surface normal.

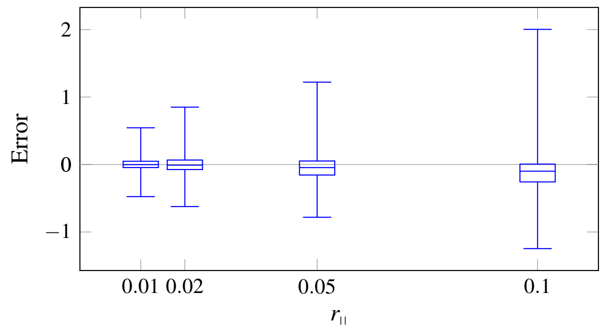


Figure 6: Box plot of error distributions for the analytical test function. We show the minimum, lower quartile, median, upper quartile and maximum errors for different values of $r_{\shortparallel}$.

gradient. As the error threshold decreases, the holes close and the stretching value approaches the ground truth solution. For the lowest error threshold value $r_{\shortparallel} = 0.01$, the number of patches at the end is still one order of magnitude smaller than the number of grid cells the function is resolved in.

6.2. Premixed Flame in a Box

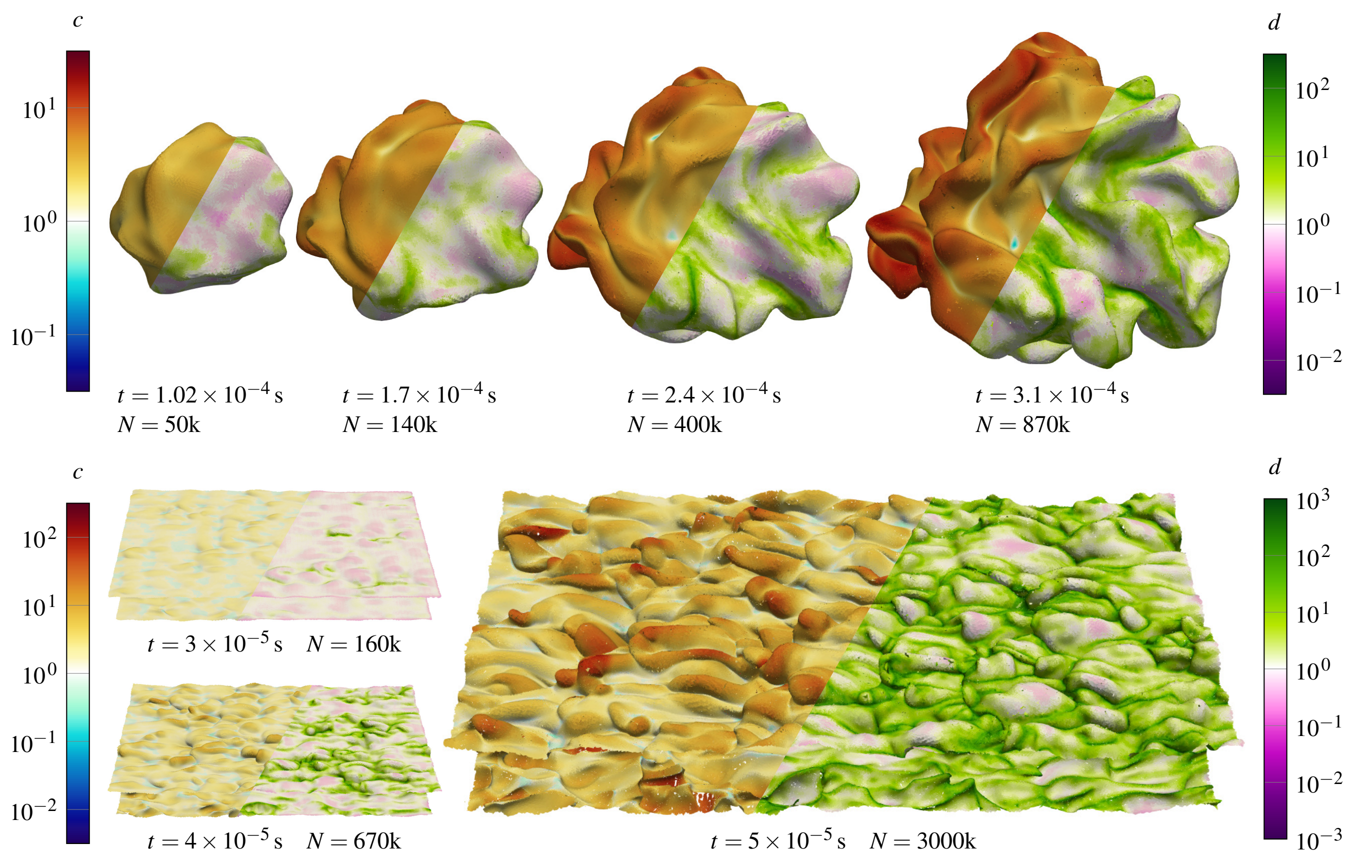
We applied our algorithm to the combustion of a premixed hydrogen-air mixture in a periodic box resolved in a $512 \times 512 \times 512$ regular grid. A high-temperature hot spot is placed in the middle of the domain, which is initialized with a flow field exhibiting isotropic turbulence. After the gas mixture is ignited, a flame front travels through the domain, consuming the fresh gas mixture and leaving burned products behind. As the flame expands, it is deformed by the turbulent flow, which is in turn influenced by the temperature and pressure changes induced by the chemical reaction. The flame surface of a premixed flame is often defined as an isosurface of the temperature between the unburnt and burnt gases, which is what we track here. Simulations of this type are relevant in safety research, where the influence of the turbulence intensity on the ignition probability of the flame is studied.

The simulation ran for about 90 h using 1024 parallel processors and performing about 21 000 iterations. We chose $r_\perp = 0.1$ and $r_{\shortparallel} = 0.02$ as well as $l_\perp = l_{\shortparallel} = 1 \times 10^{-4}$. The thresholds r_{size} and t_{size} were chosen such that a surface patch is always smaller than the smallest extent of a block of the simulation domain, and larger than $^1/_{16}$ of the size of a grid cell. Figure 7 (top) shows the results of our algorithm for four snapshots of the simulation. The simulation starts with a spherical configuration represented by about 9000 micro-patches. In the first shown time step, the surface has started to expand and wrinkle from its initial configuration. At this point, the surface is represented by about 50 000 micro-patches. Up until the last time step at $t = 3.0 \times 10^{-4}$ s, the number of patches increases to about 870 000. Despite the change in surface area and complexity, we are able to accurately track the surface without any neighbor information between micro-patches. We obtain smooth results that show which regions of the surface have expanded or contracted significantly. Figure 7 shows the change in number of patches per surface area compared to the initial seeding density. A high concentration of patches occurs in areas with high surface curvature, as well as in areas where the surface deformation has a large nonlinear component.

The ratio of the total surface area of all micro-patches to the true area of the flame surface remains stably around 3.3 for the whole simulation. This shows that our strategy for splitting and merging micro-patches is successful in maintaining a consistent and stable coverage of the surface. The number of splits performed per iteration fluctuates around 0.015 % of the total number of micro-patches at all times. The number of merge events per iteration is almost zero up until the time between the second and third snapshot shown in Figure 7, at iteration 15 000, where it starts to increase, stabilizing at around half the number of splits.

6.3. Temporal Diffusion Jet Flame

The second simulation is a temporal diffusion syngas jet flame resolved in a $1024 \times 1025 \times 512$ regular grid. In a diffusion flame, the fuel and oxidizer are not premixed before ignition, such that combustion only takes place where the two gases mix. The domain is initialized with a turbulent fuel layer in the center, surrounded by a quiescent air co-flow. The fuel layer and co-flow move in opposite directions, resulting in strong shear forces that form vortices where the two gases meet. The flame surface we track here is the isosurface of the stoichiometric mixture fraction, which is the ratio between fuel and oxidizer that theoretically results in perfect consumption of the fuel with no excess of the oxidizer.



c
10^1
10^0
10^{-1}
d
10^2
10^1
10^0
10^{-1}
10^{-2}
$t = 1.02 \times 10^{-4}$ s
$N = 50$k
$t = 1.7 \times 10^{-4}$ s
$N = 140$k
$t = 2.4 \times 10^{-4}$ s
$N = 400$k
$t = 3.1 \times 10^{-4}$ s
$N = 870$k
c
10^2
10^1
10^0
10^{-1}
10^{-2}
$t = 3 \times 10^{-5}$ s $N = 160$k
$t = 4 \times 10^{-5}$ s $N = 670$k
$t = 5 \times 10^{-5}$ s $N = 3000$k
d
10^3
10^2
10^1
10^0
10^{-1}
10^{-2}
10^{-3}

Figure 7: Results of our algorithm for the real-world simulation cases. We show the stretch coefficient c on a logarithmic scale. For the premixed flame case (top), c was computed for an interval of $\Delta t = 1.7 \times 10^{-4}$ s and $\varepsilon_{\shortparallel} = 0.02$. For the temporal diffusion jet case (bottom), we show the stretch coefficient since the start of the simulation, computed with $\varepsilon_{\shortparallel} = 0.04$. We also show the number of micro-patches per surface area as a factor of the initial density at the start of the simulation. This density factor d is also shown on a logarithmic scale.

The simulation ran for about 23 h using 4096 parallel processors. In this time, the simulation performed about 3600 iterations. For this case, we chose $r_\perp = 0.2$ and $r_{\shortparallel} = 0.04$ as well as $l_\perp = l_{\shortparallel} = 1 \times 10^{-4}$. The thresholds are chosen higher than in the premixed case, because here, we are interested in the deformation of the surface over a smaller time interval of 4×10^{-5} s. The thresholds r_{size} and t_{size} were chosen with the same method employed for the premixed case. We show our results for three different snapshots in Figure 7 (bottom). The case starts with a planar surface on both sides of the fuel jet, which is moving from left to right in the images. It is initially seeded with about 150 000 micro-patches. This number stays almost constant for the first 2000 iterations, as the surface wrinkles only a very small amount. After this, the surface starts deforming rapidly. As a result, the number of micro-patches increases rapidly up to about 3 million at the end of the simulation. The surface deformation is characterized by lots of small fuel pockets intruding into the opposing air flow. These pockets are round and smooth towards the outside, while they form a lot of sharp angles towards the inside. Both the significant expansion of surface area at the tip of these pockets as well as the sharp angles towards the fuel side are handled well by our algorithm. The ratio of the total area of all micro-patches to the true surface area remains constant here as well, but on a slightly lower level of 3.15. The number of splits is almost zero until the surface starts to deform significantly around iteration 2000. At this point, it starts to increase from 0.002 % to about 0.003 % of the total number of micro-patches until the end of the simulation. This number is much smaller than for the premixed case, because here, the surface area does not change so dramatically over time. The number of merges is initially much smaller than the number of splits, but increases steadily throughout the simulation until it is at about 75 % towards the end. This is due to the contracting behavior near the sharp angles in between fuel pockets, which causes lots of small patches to accumulate in a small area.

6.4. Performance

Figure 8 shows the performance over the course of the simulation, as well as the number of micro-patches existing at each time step. We conducted multiple experiments with different choices for the subdivision treshold $\varepsilon_{\shortparallel}$. For the sake of brevity, we only discuss the results for the lowest choice of $\varepsilon_{\shortparallel}$, which causes the highest overhead in computing time and shows the most accurate results.

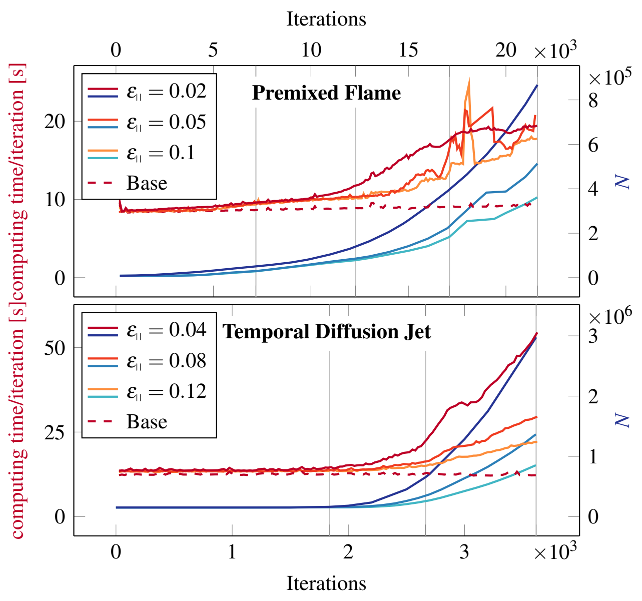


Figure 8: Computing times per iteration for different subdivision thresholds $\varepsilon_{\parallel}$. We show the base computing time of the simulation without surface tracking (- -) as well as the total times with our method enabled (☰). The number N of micro-patches over time is shown as context information (☰). The vertical lines mark the snapshots shown in Figure 7.

At the start of the premixed flame case, our implementation causes an overhead in computing time of about 300 ms per iteration, or about 3 % of the base computing time. As the flame surface expands, the overhead gradually increases, until it is at about 12.5 s or 130 % at the end of the simulation. For the temporal diffusion jet, the overhead starts out at about 1200 ms per iteration, or about 10 % of the base computing time. It remains fairly constant for the first 1500 iterations, as the surface does not change much during that time. The surface then starts to wrinkle significantly, resulting in a sharp rise of computing time, reaching up to 42 s, or about 350 % of the base time. The overhead in computing time is proportional to the number of surface patches. This means that it is also strongly dependent on the shape, area and deformation of the flame surface, which will be different for each simulation case. For the premixed flame case, our implementation generates an overhead of about 30 h or 50 % for the whole simulation. The memory consumption behaves very similarly, as it is also directly dependent on the number of micro-patches. At the start of the simulation, we consume about 90 GB of additional memory, which is an overhead of about 30 % for this simulation case. This rises to about 400 GB at the end of the simulation, which represents an overhead of about 130 %. The total overhead in computing time caused by the temporal jet case is 8.4 h or about 67 %. The additional memory consumption here reaches from 800 GB to 1800 GB, or 44 % to 150 %.

If we were to implement our approach as a post-processing step, we would need to store the raw simulation data to disk, possibly transfer it over a network, and read it again for each time step. If we only store the simulation variables that are directly needed for tracking the flame surface (flow velocity and one scalar variable), we would already need 100 TB of storage space for the whole premixed flame case, and 60 TB for the temporal jet case. This is more than is typically available for a user or project on a current high performance computing cluster. Additionally, writing this data to disk for every simulation time step alone would incur about 21 h of run time on our computing cluster for the premixed case and about 13 h for the temporal jet case. If we write all simulation variables, this increases to 420 TB (90 h) and 320 TB (70 h), respectively. Considering that this is just part of the overhead inherent to a post-processing approach makes it clear that an on-the-fly solution is the only way of obtaining results with reasonable effort.

7. Discussion

Due to its nature as an algorithm designed for on-the-fly execution in a highly parallel environment, our method has some inherent limitations. We can not go back in time to refine micro-patches before errors due to insufficient resolution occur. This means that we have no way of correcting for the accumulation of errors in the estimation of the tangential deformation that will inevitably occur, except for splitting our micro-patches whenever the error becomes too large. This leads to a large number of micro-patches over time, which rises faster than the surface area of the flame itself. Therefore, the error thresholds $r_{\perp}$ and $r_{\parallel}$ have to be carefully chosen, taking into account the acceptable error for the maximum investigated time interval, and the overhead in terms of memory consumption and computing time added to the simulation. This is not a trivial task and requires some experience of the user, and it is not clear how it could be simplified. However, the tangential deformation will generally be investigated in statistics with other simulated quantities, where the error can be analyzed and taken into account. In future work, we want to investigate a strategy for better controlling the number of micro-patches over time. This could be handled by periodically merging or redistributing patches on the surface. This is a global process that requires a common parameterization of the surface for micro-patches in a local neighborhood.

Because we track independent micro-patches, we do not produce a closed, manifold mesh of the flame surface. This is due to our strict parallelization requirements, which are not met by the global nature of subdivision and join operations in meshes. Direct rendering of the data produced by our method can be done via splatting of the micro-patches. If a closed mesh surface is required, it can be reconstructed using any available meshing algorithm for point clouds.

Because our algorithm only tracks micro-patches initialized on the starting surface, it does not handle the case of new disconnected surface parts appearing during the course of the simulation. This can be easily addressed by periodically checking for new parts of the surface that are not yet covered, and initializing new patches. For new surface parts streaming in from a non-periodic boundary, a more sophisticated approach might be necessary, which is a subject for future work.

If we wanted to measure the tangential deformation of the surface only at single points, we could simply compute the product integral of the instantaneous Jacobian $\nabla\mathbf{u}$ along the path of each point. This would not require the tracking of ghost particles and would be cheaper in terms of communication and memory overhead. However, our goal is to track the whole surface. By measuring deformation based on the relative movement of multiple points, we get the average of a finite part of the surface. This introduces a filtering effect and better represents the behavior of the surface as a whole. More importantly though, we need the information gained from tracking a group of points to compute the error measures $e_{\shortparallel}$ and $e_{\perp}$, which are based on the deviation from linear behavior. Without this information, which is not included in $\nabla \mathbf{u}$, we could only use more inaccurate criteria for splitting and merging patches. This would lead to larger errors in the measurement of deformation and larger holes in the surface.

Our algorithm is the first to provide a viable way of tracking the whole flame surface on-the-fly over the complete simulation time. This opens new pathways to the investigation of flame behavior. The single snapshots that were routinely observed before did not show detailed changes in surface shape over time, and the correspondence between points on the surface in two different snapshots could not be reconstructed. Because we explicitly track single points on the surface over extended periods of time, the evolution of simulation variables at the point positions over time also becomes easily observable. Obtaining this data for the complete flame surface enables visual and statistical evaluation of direct numerical combustion simulations on a new scale. Our novel method of measuring tangential deformation provides combustion researchers with a new quantity to study the effects of flame-turbulence interactions. This integration-based quantity is only made possible by the on-the-fly nature of the algorithm. Accurate path line integration is simply not possible as a post process, if the simulation data can only be stored to disk in a massively reduced temporal or spatial resolution.

The overhead in memory and computing time caused by our method is fairly large compared to existing on-the-fly visualization approaches, which are generally designed to require only a small fraction of the computing time of the simulation [Ma09]. In this context, it is important to note that our approach is not meant to be a visualization method only, and it is not meant to be a general-purpose tool that is activated for every simulation. It is a specialized analysis tool that can be used in situations where combustion researchers are specifically interested in the detailed behavior of the flame surface over long time periods. Such tools are necessary when a detailed analysis of the low-level behavior of the flame is required. For example, Scholtissek et al. [SDGH17] report a four-fold increase in computing time for their gradient trajectory tracking, which enabled them to develop a more accurate flamelet model. In this case, the analysis was employed in the context of a research project that required accurate low-level information which can only be achieved with high computational overhead. We expect our algorithm to be used in a similar context. It is of particular interest for the building of combustion models and the investigation of local flame extinction and reignition mechanics. Existing combustion literature, such as works by Sripakagorn et al. [SMKP04], observe the history of temperature and heat release at single points on the surface that are tracked over time. By providing such histories for a great number of points covering the whole flame surface, we enable a statistical evaluation leading to new models for flame behavior. The local tangential deformation of the flame surface over extended time intervals has not been extensively studied in combustion literature.

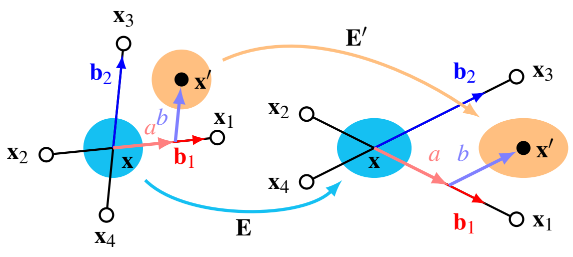


Figure 9: Interpolating the transformation at an offset position $\mathbf{x}'$. The ghost particles in the direction of $\mathbf{x}'$ have been stretched more than their counterparts during the time interval. Therefore the interpolated $\mathbf{E}'$ has a stronger stretching effect on the local neighborhood of $\mathbf{x}'$ (●) than $\mathbf{E}$ stretches the local neighborhood of $\mathbf{x}$ (●).

We believe it has a significant impact on flame behavior, as rapid tangential stretching or compression of the gases in the combustion zone will change the local conditions. With our algorithm, our collaboration partners will be able to test this hypothesis and gain a deeper understanding of the effects of tangential deformation on turbulent combustion than previously possible.

Acknowledgements

This work was supported by DFG grant no. TH 693/12. 3D figures were generated using ParaView [AGL*05].

Appendix A: Interpolating the Transformation for new Patches

If the deformation across the parent patch was completely uniform, we could just pass this deformation on to the child patches after a split. However, in general there will be slight differences in the transformations each of the four ghost particles have experienced. At the time of the split, we still have this information for the time interval since the parent patch was last reset (or first created).

Let $\mathbf{x}_i(t_{k-1})$ be the positions of the ghost particles (relative to the central point) at the last reset of the parent patch, or at the start time t_s if the patch was not reset since that time. Let $\mathbf{x}_i(t_k)$ be the ghost particle positions at the time of the split. Then $\mathbf{E}_{t_{k-1}}^{t_k}$ obtained via (3) can be thought of as the solution of the system

$$\begin{pmatrix} \mathbf{b}_1(t_{k-1})^{\mathrm{T}} \\ \mathbf{b}_2(t_{k-1})^{\mathrm{T}} \\ \mathbf{n}(t_{k-1})^{\mathrm{T}} \end{pmatrix} \left(\mathbf{E}_{t_{k-1}}^{t_k}\right)^{\mathrm{T}} = \begin{pmatrix} \mathbf{b}_1(t_k)^{\mathrm{T}} \\ \mathbf{b}_2(t_k)^{\mathrm{T}} \\ \mathbf{n}(t_k)^{\mathrm{T}} \end{pmatrix}$$

$$\mathbf{b}_1(t) = (\mathbf{x}_1(t) - \mathbf{x}_2(t))/2$$
$$\mathbf{b}_2(t) = (\mathbf{x}_3(t) - \mathbf{x}_4(t))/2\,.$$

In other words, $\mathbf{E}_{t_{k-1}}^{t_k}$ is the average of the transformations that map the corresponding ghost particles to each other exactly.

For the central point, it makes sense to weight those transformations equally, but if we want to initialize a new patch whose center is slightly offset, we get a more accurate result if we adjust the weights depending on where the new center is located. To this effect, we parameterize the tangent space of the micro-patch by expressing it as a linear combination of the two base vectors $\mathbf{b}_1(t)$ and $\mathbf{b}_2(t)$. We can now express the location of the new center $\mathbf{x}'$ in this new basis:

$$\mathbf{x}' = \lambda_1 \mathbf{b}_1(t_k) + \lambda_2 \mathbf{b}_2(t_k)\,.$$

These coordinates correspond directly to the coordinates of $(\mathbf{E}_{t_{k-1}}^{t_k})^{-1}\mathbf{x}'$ at time t_{k-1}:

$$(\mathbf{E}_{t_{k-1}}^{t_k})^{-1}\mathbf{x}' = \lambda_1 \mathbf{b}_1(t_{k-1}) + \lambda_2 \mathbf{b}_2(t_{k-1})\,.$$

If λ_1 is positive, i.e., if $\mathbf{x}'$ is located more towards $\mathbf{x}_1(t_k)$ than towards $\mathbf{x}_2(t_k)$, we want $\mathbf{x}_1(t_k)$ to have a stronger influence on the result. The same applies to the direction of $\mathbf{b}_2$. We therefore compute new interpolated base vectors $\mathbf{b}'_{1,2}(t)$ by weighting the ghost particle positions with λ_1 and λ_2:

$$\begin{aligned}\mathbf{b}'_1(t) &= (1+2\lambda_1)\,\mathbf{x}_1(t) - (1-2\lambda_1)\,\mathbf{x}_2(t)\\ \mathbf{b}'_2(t) &= (1+2\lambda_2)\,\mathbf{x}_3(t) - (1-2\lambda_2)\,\mathbf{x}_4(t)\,.\end{aligned}$$

The adjusted transformation $\mathbf{E}_{t_{k-1}}^{t_k}{}'$ is then the solution to the system

$$\begin{pmatrix}\mathbf{b}'_1(t_{k-1})^{\mathrm{T}}\\ \mathbf{b}'_2(t_{k-1})^{\mathrm{T}}\\ \mathbf{n}(t_{k-1})^{\mathrm{T}}\end{pmatrix}\left(\mathbf{E}_{t_{k-1}}^{t_k}{}'\right)^{\mathrm{T}} = \begin{pmatrix}\mathbf{b}'_1(t_k)^{\mathrm{T}}\\ \mathbf{b}'_2(t_k)^{\mathrm{T}}\\ \mathbf{n}(t_k)^{\mathrm{T}}\end{pmatrix}.$$

The complete transformation of a micro-patch that has been split off from a parent at some intermediate time t_k is then obtained by

$$\mathbf{E}_{t_s}^{t_e} = \mathbf{E}_{t_n}^{t_e}\cdot\mathbf{E}_{t_{n-1}}^{t_n}\cdot\cdots\cdot\mathbf{E}_{t_k}^{t_{k+1}}\cdot\mathbf{E}_{t_{k-1}}^{t_k}{}'\cdot\mathbf{E}_{t_s}^{t_{k-1}}\,.$$

Here, $\mathbf{E}_{t_s}^{t_{k-1}}$ is the transformation of the parent patch from the start of the interval up to the time when its ghost particles were last reset before the split operation at t_k. If $t_s > t_{k-1}$, it is omitted. Here, $\mathbf{E}_{t_s}^{t_{k-1}}$ is the transformation of the parent patch from the start of the interval up to the time when its ghost particles were last reset before the split operation at t_k. $\mathbf{E}_{t_{k-1}}^{t_k}{}'$ is the estimated transformation at a point with a slight offset from the center of the parent patch in the interval before the split. $\mathbf{E}_{t_{k-1}}^{t_k}{}'$ is the estimated transformation at a point with a slight offset from the center of the parent patch in the interval before the split.

References

[ACG*14] Agranovsky A., Camp D., Garth C., Bethel E. W., Joy K. I., Childs H.: Improved post hoc flow analysis via lagrangian representations. In *IEEE LDAV* (2014), pp. 67–75. 2

[AFO*16] Abdelsamie A., Fru G., Oster T., Dietzsch F., Janiga G., Thévenin D.: Towards direct numerical simulations of low-mach number turbulent reacting and two-phase flows using immersed boundaries. *Computers & Fluids 131* (2016), 123 – 141. 2, 5

[AGL*05] Ahrens J., Geveci B., Law C., Hansen C., Johnson C.: Paraview: An end-user tool for large-data visualization. In *The Visualization Handbook*. Elsevier, 2005. 10

[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

[BFTW09] Bürger K., Ferstl F., Theisel H., Westermann R.: Interactive streak surface visualization on the GPU. *IEEE TVCG (Proc. VIS) 15*, 6 (2009), 1259–1266. 2

[BOJH15] Berres A., Obermaier H., Joy K., Hagen H.: Adaptive particle relaxation for time surfaces. In *IEEE PacificVis* (2015), pp. 147–151. 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 TVCG 16*, 2 (2010), 248–260. 2

[CA97] Crossno P., Angel E.: Isosurface extraction using particle systems. In *Proc. 8th Conference on Visualization* (1997), IEEE, pp. 495–ff. 2

[CCG*12] Camp D., Childs H., Garth C., Pugmire D., Joy K. I.: Parallel stream surface computation for large data sets. In *IEEE LDAV* (2012), pp. 39–47. 2

[CMN13] Clyne J., Mininni P., Norton A.: Physically-based feature tracking for cfd data. *IEEE TVCG 19*, 6 (2013), 1020–1033. 2

[DHS*12] Duque E. P., Hiepler D., Stone C. P., Legensky S. M., Ma K.-L., Muelder C., Wei J.: Ifdt – intelligent in-situ feature detection, extraction, tracking and visualization for turbulent flow simulations. In *7th International Conference on Computational Fluid Dynamics* (2012). 2

[GTS04] Garth C., Tricoche X., Scheuermann G.: Tracking of vector field singularities in unstructured 3D time-dependent datasets. In *IEEE TVCG (Proc. VIS)* (2004), pp. 329–336. 2

[GWT*08] Garth C., Wiebel A., Tricoche X., Joy K., Scheuermann G.: Lagrangian visualization of flow-embedded surface structures. In *Computer Graphics Forum* (2008), vol. 27, Wiley Online Library, pp. 1007–1014. 2

[Hal02] Haller G.: Lagrangian coherent structures from approximate velocity data. *Physics of Fluids 14*, 6 (2002), 1851–1861. 6

[KGJ09] Krishnan H., Garth C., Joy K.: Time and streak surfaces for flow visualization in large time-varying data sets. *IEEE TVCG 15*, 6 (2009), 1267–1274. 2

[Ma09] Ma K.-L.: In situ visualization at extreme scale: Challenges and opportunities. *IEEE CGA 29*, 6 (2009), 14–19. 10

[MGYC09] Mascarenhas A., Grout R., Yoo C., Chen J.: Tracking flame base movement and interaction with ignition kernels using topological methods. In *Journal of Physics: Conference Series* (2009), vol. 180, IOP Publishing, p. 012086. 2

[MM09] Muelder C., Ma K.-L.: Interactive feature extraction and tracking by utilizing region coherency. In *IEEE PacificVis* (2009), IEEE, pp. 17–24. 2

[MPI15] MPI Forum: *MPI: A Message-Passing Interface Standard, Version 3.1*. High Performance Computing Center Stuttgart (HLRS), 2015. 5

[PV12] Poinsot T., Veynante D.: *Theoretical and Numerical Combustion*. T. Poinsot and D. Veynante, 2012. 1, 2

[RRTC99] Renard P.-H., Rolon J. C., Thévenin D., Candel S.: Investigations of heat release, extinction, and time evolution of the flame surface, for a nonpremixed flame interacting with a vortex. *Combustion and Flame 117*, 1–2 (1999), 189–205. 2

[SDGH17] Scholtissek A., Dietzsch F., Gauding M., Hasse C.: In-situ tracking of mixture fraction gradient trajectories and unsteady flamelet analysis in turbulent non-premixed combustion. *Combustion and Flame 175* (2017), 243–258. 2, 10

[SMKP04] Sripakagorn P., Mitarai S., Kosály G., Pitsch H.: Extinction and reignition in a diffusion flame: a direct numerical simulation study. *Journal of Fluid Mechanics 518* (Nov 2004), 231–259. 1, 2, 10

[SX97] Silver D., X.Wang: Tracking and visualizing turbulent 3D features. *IEEE TVCG 3*, 2 (1997), 129–141. 2

[SYM14] Sauer F., Yu H., Ma K.-L.: Trajectory-based flow feature tracking in joint particle/volume datasets. *IEEE TVCG 20*, 12 (Dec 2014), 2565–2574. 2

[TS03] Theisel H., Seidel H.-P.: Feature flow fields. In *Proc. VisSym* (2003), pp. 141–148. 2

[WYM13] Wang Y., Yu H., Ma K.-L.: Scalable parallel feature extraction and tracking for large time-varying 3d volume data. In *EGPGV* (2013), pp. 17–24. 2

[YWG*10] Yu H., Wang C., Grout R. W., Chen J. H., Ma K.-L.: In situ visualization for large-scale combustion simulations. *IEEE CGA 30*, 3 (2010), 45–57. 2