Locally Adapted Reference Frame Fields using Moving Least Squares

Julio Rey Ramirez, Peter Rautek, Tobias Günther, and Markus Hadwiger

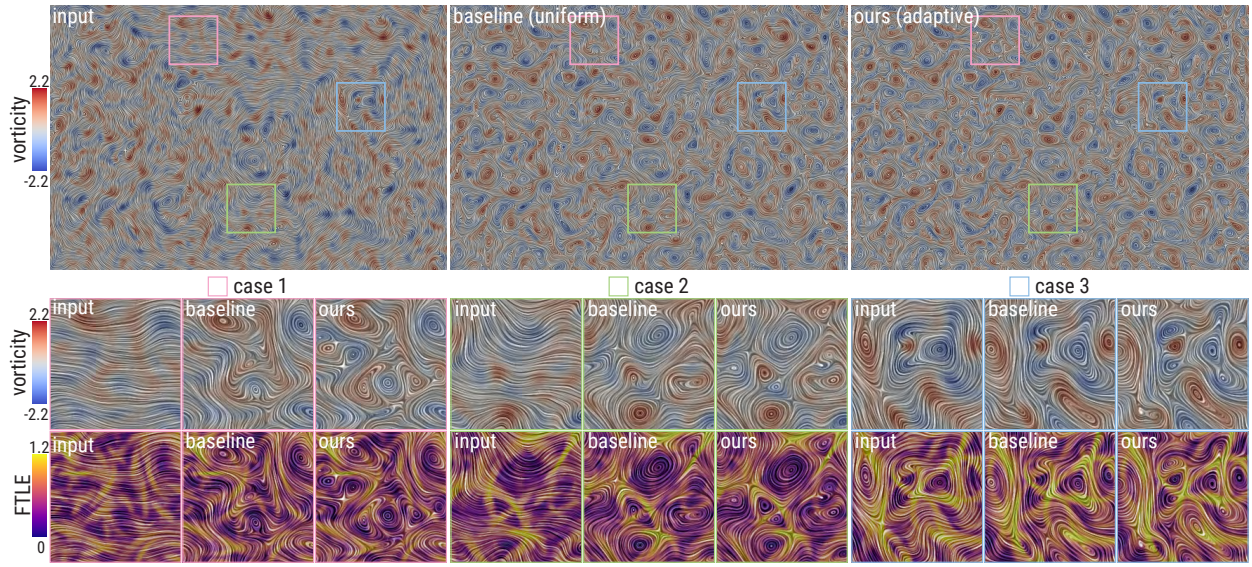

Fig. 1: Turbulent flow contains many small-scale vortices that are separated by transport barriers (FTLE). Prior reference frame optimizations [13] (baseline) cannot resolve the details, since reference frames are optimized for uniformly weighted neighborhoods. With our method, the neighborhood weighting is guided by FTLE, allowing the detection of small-scale vortices.

Abstract—The detection and analysis of features in fluid flow are important tasks in fluid mechanics and flow visualization. One recent class of methods to approach this problem is to first compute objective optimal reference frames, relative to which the input vector field becomes as steady as possible. However, existing methods either optimize locally over a fixed neighborhood, which might not match the extent of interesting features well, or perform global optimization, which is costly. We propose a novel objective method for the computation of optimal reference frames that automatically adapts to the flow field locally, without having to choose neighborhoods a priori. We enable adaptivity by formulating this problem as a moving least squares approximation, through which we determine a continuous field of reference frames. To incorporate fluid features into the computation of the reference frame field, we introduce the use of a scalar guidance field into the moving least squares approximation. The guidance field determines a curved manifold on which a regularly sampled input vector field becomes a set of irregularly spaced samples, which then forms the input to the moving least squares approximation. Although the guidance field can be any scalar field, by using a field that corresponds to flow features the resulting reference frame field will adapt accordingly. We show that using an FTLE field as the guidance field results in a reference frame field that adapts better to local features in the flow than prior work. However, our moving least squares framework is formulated in a very general way, and therefore other types of guidance fields could be used in the future to adapt to local fluid features.

Index Terms—Scientific visualization, unsteady flow, reference frame optimization.

1 Introduction

In the real world, the intricate dynamics of fluids under motion are notoriously difficult to comprehend. Only through flow visualization can vortices, jets, and barriers become visible, for example, through the injection of smoke or dye. In computational fluid dynamics, flow visualization plays an equally important role, enabling a deeper analysis of transport, settling, and mixing. Currently, the complexity of time-dependent fluids still requires the development of novel approaches that allow the effective exploration of coherent structures and their evolution [6]. A key requirement in this endeavor is that analysis approaches should always lead to the same conclusions, regardless of how the fluid flow is observed. This led to the definition of a formal property, called objectivity [53], which refers to the invariance of a measure under temporally smooth rotations and translations of the reference frame. One such objective approach is the search for a field of new observers that move with the structures in the fluid, such that their apparent flow becomes as stationary as possible [13, 20].

However, existing local reference frame optimization approaches have made one key assumption: To calculate the optimal observer at a given location, these methods assumed that all flow structures move coherently within a finite-sized neighborhood centered at the

- *Julio Rey Ramirez, Peter Rautek, and Markus Hadwiger are with King Abdullah University of Science and Technology (KAUST), Thuwal, 23955-6900, Saudi Arabia. E-mail: {julio.reyramirez | peter.rautek | markus.hadwiger}@kaust.edu.sa.*
- *Tobias Günther is with the Department of Computer Science at Friedrich-Alexander-Universität Erlangen-Nürnberg. E-mail: tobias.guenther@fau.de*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

location. The neighborhood was typically assumed to be rectangular or circular, and its size effectively determines the resolution at which fluid structures can be resolved. Consequently, the approach has been applied mainly to fluid flows with a low to moderate Reynolds number (Re < 320) [2], which in fluid dynamics is considered laminar flow. The more turbulent the flow, the higher the likelihood that nearby small-scale vortices move in different directions, which cannot be captured by prior work.

In this paper, we propose the first reference frame optimization that is specifically targeted at 2D *turbulent* flow. To this end, we present a novel adaptive neighborhood weighting that adaptively adjusts to the transport barriers of the flow field. The transport barriers, known as hyperbolic Lagrangian coherent structures (LCS), are approximated by calculating the finite-time Lyapunov exponent (FTLE) [48] in forward and backward time, which gives rise to a scalar field. Using FTLE, we derive a guidance field that is sampled and lifted to obtain a curved manifold adapted to local flow features, giving rise to valleys and plateaus corresponding to coherent flow regions, and ridges acting as barriers separating different regions.

We formulate the reference frame optimization as a general moving least squares (MLS) approximation problem, which provides a strong mathematical foundation for our approach. The guidance field is used to provide a distance measure for the MLS approximation. This distance determines the weight for the neighborhood samples during frame optimization at a given point, adjusting to one coherent structure at a time. This idea is illustrated in Fig. 2. Prior work assumed a coherent flow motion within the neighborhood (uniform weighting) during local optimization, which prevents existing techniques from detecting small-scale vortices moving in different directions. Our technique adapts to the coherent region around a given point by adjusting the weighting with the distances provided by the guidance field. With this, we lay the foundation for the application of reference frame optimization in a vast number of fields, ranging from engineering (aerospace, turbomachinery) [45], atmospheric sciences (hurricane and ocean eddy genesis) [9], to biomedical applications (hemodynamics in aneurysms) [38] and more. Across all these fields, the ability to extract vortex coreline geometry is paramount as it enables a quantitative study of the fluid dynamics, including the genesis and lifetime of vortices, as well as their co-location with structures that they dynamically interact with.

As shown in Fig. 1, our novel reference frame optimization using moving least squares detects small co-rotating vortex pairs that are undetected by existing reference frame optimization methods. We evaluate our method for both laminar and turbulent fluid flows.

2 Related Work

2.1 Feature Extraction in Unsteady Flow

Time-dependent flows are complex spatially and temporally varying dynamical systems whose evolution is governed by fluid flow structures, such as vortices, jets, and barriers. In fluid mechanics, Lagrangian coherent structures (LCS) [24] describe boundaries of coherent motion in the fluid. The term Lagrangian indicates that those coherent structures are formed from pathlines, i.e., a pathline is either part of an LCS or it may not cross an LCS, which gives rise to a domain partition into coherent regions. Coherent regions can also be formed by so-called coherent sets [11], which are sets of trajectories with similar behavior. In the literature, three types of LCS are commonly distinguished: *elliptic LCS* describe the boundaries of vortices, for example, through inhibition of vorticity diffusion, *parabolic LCS* denote the centerline of jets, which exhibit minimal shear within a regime of otherwise shearing flow, and lastly, *hyperbolic LCS* act as attracting and repelling fluid structures that share some resemblance with separatrices known from the topological analysis of stationary flow [55]. In fact, the extension from stationary to time-dependent vector field topology was subject to a significant amount of research. Critical points (also known as stationary points) [25,26] have been tracked [59], for example, with feature flow fields [51], which turn out to be estimators of Galilean vortex transport [2]. The paths of vortices have been found as maxima in vorticity, locations of vanishing acceleration [30], and critical points in stationary reference frames [2], to name a few. The paths of saddles have been found through the intersection of hyperbolic LCS, and locally via distinguished hyperbolic trajectories (DHTs) [5,27,28]. In streakline-based vector field topology [47,54], streaklines, emanating from intersections of hyperbolic LCS, are the counterpart to separatrices of stationary flow. Within the flow visualization literature, further interesting fluid structures have been identified, such as recirculations [58] or attachments/detachments through splats and anti-splats [40]. For further discussions, we refer the reader to recent surveys on topology-based and feature-based flow analysis [6,12,16].

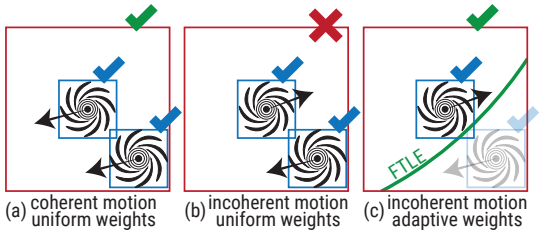

Fig. 2: Illustration of local reference frame optimization using uniform and adaptive weighting. Neighborhoods enclosing a single vortex (blue) lead to successful optimization for both uniform and adaptive weighting. While uniform weighting in neighborhoods enclosing multiple vortices (red) works for coherent vortex motion (a), it fails if vortices move in different directions (b). Our adaptive weighting constrains the optimization to coherent regions limited by the transport barriers (green line).

Finite-time Lyapunov Exponent (FTLE) fields. In flow visualization, Lagrangian coherent structures (LCS) are commonly approximated as *ridges* of the finite-time Lyapunov exponent (FTLE) field [48], which measures the separation of nearby-released tracer particles after a finite time interval T as

$$\sigma_{t_0}^{T}(x) = \frac{1}{|T|} \ln \sqrt{\lambda_{max}(\mathbf{C}(x))}. \tag{1}$$

Here, t_0 is the initial time for the integration, T (positive or negative) is the integration time, and $\lambda_{max}(\mathbf{C})$ is the largest eigenvalue of the right Cauchy-Green deformation tensor $\mathbf{C}$, computed from the gradient of the flow map at x. For a vector field $\boldsymbol{v}$, let $t \mapsto x(t)$ be a pathline

$$x(t) = x_0 + \int_{t_0}^{t} \boldsymbol{v}(x(\tau),\tau)\ \mathrm{d}\tau, \tag{2}$$

with initial position $x(t_0) = x_0$. Then, the flow map and its gradient are

$$\phi_{t_0}^{T}(x_0) := x(t_0+T), \qquad \mathbf{F} = \frac{\partial \phi_{t_0}^{T}(x_0)}{\partial x_0}. \tag{3}$$

The Cauchy-Green tensor is then given by $\mathbf{C} = \mathbf{F}^T\mathbf{F}$ [48]. To simplify the notation, for given t_0, T, we denote the FTLE field as $\sigma(x)$. The flow map gradient $\mathbf{F}$ has been approximated by finite differences [22], and locally through linearization of the flow map via matrix exponentiation of the Jacobian [29]. We refer to Kuhn et al. [33] for a benchmark of computation methods. FTLE fields have been computed adaptively [4,46], interactively [3], and were rendered using a Monte Carlo method [1,14]. The extraction of FTLE ridge lines has been approached by tracking through space-time [49,57].

2.2 Reference Frame Optimization

In recent work on flow analysis, visualization, and feature extraction in unsteady flow, methods that are invariant to the choice of reference frame have become increasingly important. Early approaches focused on *Galilean invariance* (i.e., invariance under equal-speed translations of the reference frame) to account for uniform background motion. *Objectivity*, introduced in continuum mechanics by Truesdell and Noll [53] and further elaborated by Haller [23], formalizes the requirement that flow features remain the same under *any* smooth time-dependent rigid transformation of the observer. In an objective setting, transformations

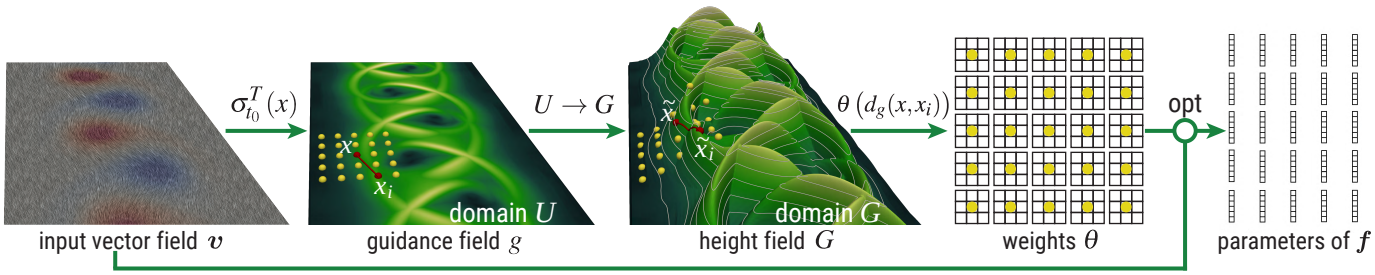

Fig. 3: **Pipeline overview.** Given a 2D unsteady input flow field $\boldsymbol{v}(x,t),\ x \in U$, we compute the FTLE scalar field $\sigma^T_{t_0}(x)$ and use it as a guidance field g (Eq. 19). The guidance field defines a height field manifold G (Eq. 20), on which the straight line between two grid points $x, x_i \in U$ (red line) becomes the curved arc induced on the height field G between the points $\tilde{x}, \tilde{x}_i \in G$ (red arc). The induced distances d_g (Eq. 21), defined as the induced arc length, determine the weights θ (Eq. 15) for the moving least squares optimization (*opt*) (Eq. 25), which automatically adapts the local neighborhoods to the Lagrangian coherent structures of the input flow field, producing a locally adapted smooth reference frame field $x \mapsto \boldsymbol{f}(x),\ x \in U$.

include both translations and rotations that may vary over time. Consequently, feature definitions that are *objective* do not depend on how the flow is observed, making them better suited for analyzing complex unsteady fluid phenomena. Vortex definitions that are not objective may yield inconsistent or observer-dependent results.

Classic work on vortex criteria [23, 37] highlighted that purely Galilean-invariant definitions can fail for more general observer motions, whereas purely rotating frames (e.g., rotation-invariant [15]) can be insufficient if the flow itself has more complex motion. A powerful approach for objective reference frame optimization is to find a suitable time-dependent local reference frame that minimizes the unsteadiness of the observed flow. Günther et al. [13] proposed a local transformation at each point such that the time derivative of the flow is minimized. The solution in a local neighborhood forms the *observed* field. This yields objective vortex measures since both velocity and its derivatives become frame-independent [2, 13]. Kaszás et al. [31] optimize for the observer that is most similar to the input flow field in the L_2 norm. Local optimization has also been exploited to make classical criteria (such as λ_2, vorticity extrema, or Sujudi-Haimes) objective [13, 50]. Global solutions for nearly-rigid fields of observers, given by a global observer velocity field, have been proposed [20, 43].

Further generalizations increased the degrees of freedom [2, 18], modeled the motion of finite-sized particles [17], restricted observers to follow pathlines [19], and fitted observers to sets of trajectories [10]. Using machine learning, reference frames have been optimized non-objectively with CNNs [32] and objectively with transformers [61].

The extraction of rigid reference frames for interactive visualization and vortex extraction has been presented [44, 60], and local reference frames have been used to improve the extraction of LCS in 2D unsteady flow [62]. The starting point of our method is the reference frame optimization method proposed by Günther et al. [13]. This method estimates a local reference frame transformation for each point in the domain, such that the observed field is as steady as possible, i.e., the norm of the transformed time derivative is minimized. For each point, the minimization is computed in a neighborhood around that point, assuming that the reference frame transformations are constant in this neighborhood. One drawback of this approach is that all points in the pre-defined neighborhood have an equal contribution to the solution. We extend this method to allow for adaptive neighborhoods by formulating the problem as a moving least-squares approximation, with a carefully designed distance measure determined by a scalar guidance field that separates the fluid domain into coherent regions.

2.3 Moving Least Squares (MLS) Approximation

The properties and approximation power of moving least squares (MLS) methods for function approximation and interpolation were originally investigated by Lancaster and Salkauskas [34], and extended in many follow-up works, for example by Levin [35]. In particular, the stability and robustness of the basic method have been investigated and improved, such as in robust MLS [8] or stable MLS [36]. Introductions are given in the books by Fasshauer [7] and Wendland [56], respectively, as well as in the short overview introduction by Nealen [39].

MLS computes a smooth approximating function that results from performing a local polynomial fitting in the weighted least-squares sense at each output position x. Given parameters $\boldsymbol{p}$, determining a polynomial in several variables, the optimal parameters $\boldsymbol{p}(x)$ are computed using a local weighted least squares fit at each point x, approximating an input function known only at discrete data sites (samples) x_i, with corresponding function values f_i. The approximating function $f(x)$ is determined at each x by optimal parameters $\boldsymbol{p}(x)$. To emphasize the parameter dependence, we can write the approximating function as $f_{\boldsymbol{p}}(x)$. At each point x, MLS solves for the optimal parameters

$$\boldsymbol{p}(x) = \operatorname{argmin}_{\boldsymbol{p}'} \sum_{i \in I(x)} \left(f_{\boldsymbol{p}'}(x_i) - f_i\right)^2 \Theta(x, x_i)\,. \tag{4}$$

The local weighted least squares approximation considers all samples from an index set $I(x)$, and the weight of each sample x_i is determined by a weighting function Θ. The properties of Θ determine the smoothness of the global output function $f(x)$, corresponding to the parameters $\boldsymbol{p}(x)$. Often, $\Theta(x,x_i) = \theta(\|x - x_i\|)$, and the index set $I(x)$ depends on the position x, e.g., it contains all indices i such that $\|x - x_i\| < r \in \mathbb{R}$.

3 Method

The input to our method is an unsteady flow field $\boldsymbol{v}(x,t)$, for which we compute a smooth field of optimal reference frames $\boldsymbol{f}(x,t)$. An overview of the approach is given in Fig. 3. First, the FTLE field $\sigma^T_{t_0}(x)$ is calculated, which is used to define a height field. The goal is to compute the reference frame only from sample points that are on the same side of a ridge. To determine whether a sample point is on the other side of a ridge, we integrate the walking distance $d_g(x,x_i)$ from the center point x of the neighborhood window to the sample point x_i, which is then mapped via a transfer function to a weight θ that is included into the reference frame optimization. As in previous work [13] on optimal reference frames, we compute our optimal frame field for each timestep t individually, and thus we omit time t to simplify the notation $\boldsymbol{f}(x) := \boldsymbol{f}(x,t)$ assuming the time t is known from context. The time parameter t should be understood to be included implicitly.

3.1 Reference Frame Fields

For a given 2D unsteady input vector field $\boldsymbol{v}$, given on the spatial domain $U \subset \mathbb{R}^2$, our goal is to compute a smooth *reference frame field* $\boldsymbol{f}$. We define the field $\boldsymbol{f}$ as a smooth 6D parameter vector field

$$\begin{aligned} \boldsymbol{f} &: U \to \mathbb{R}^6, \\ & x \mapsto \boldsymbol{f}(x). \end{aligned} \tag{5}$$

The frame parameters $\boldsymbol{f}(x)$, at a point $x \in U$, are given by a vector

$$\boldsymbol{f}(x) = \begin{pmatrix} a & b & c & \dot{a} & \dot{b} & \dot{c} \end{pmatrix}^T (x). \tag{6}$$

Following Zhang et al. [60], these six parameters determine a reference frame as a Killing vector field $\boldsymbol{w}_{\boldsymbol{f}}(y)$ on U, $y \in U$, and its corresponding Eulerian time derivative $\partial \boldsymbol{w}_{\boldsymbol{f}}(y)/\partial t$, via the linear combinations

$$\boldsymbol{w}_{\boldsymbol{f}}(y) = a\,\boldsymbol{e}_1(y) + b\,\boldsymbol{e}_2(y) + c\,\boldsymbol{e}_3(y), \tag{7}$$

$$\frac{\partial \boldsymbol{w}_{\boldsymbol{f}}}{\partial t}(y) = \dot{a}\,\boldsymbol{e}_1(y) + \dot{b}\,\boldsymbol{e}_2(y) + \dot{c}\,\boldsymbol{e}_3(y). \tag{8}$$

In the above, the subscript $\boldsymbol{f}$ denotes that the Killing field $y \mapsto \boldsymbol{w}_{\boldsymbol{f}}(y)$ is completely determined by a single given parameter vector $\boldsymbol{f} \in \mathbb{R}^6$.

As Zhang et al. [60], we use the three basis Killing fields given by

$$\boldsymbol{e}_1(y) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \;,\; \boldsymbol{e}_2(y) = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \;,\; \boldsymbol{e}_3(y) = \begin{pmatrix} -(\hat{\mathrm{y}}(y) - \hat{\mathrm{y}}_m) \\ (\hat{\mathrm{x}}(y) - \hat{\mathrm{x}}_m) \end{pmatrix}. \tag{9}$$

Here, $(\hat{\mathrm{x}}, \hat{\mathrm{y}})(y)$ are the Cartesian coordinates of the point $y \in U$, and the coordinates $(\hat{\mathrm{x}}_m, \hat{\mathrm{y}}_m)$ refer to the center of mass of the domain U.

From Eqs. 7 and 9, we also immediately get the velocity gradient

$$\nabla \boldsymbol{w}_{\boldsymbol{f}}(y) = \begin{pmatrix} 0 & -c \\ c & 0 \end{pmatrix}. \tag{10}$$

It is crucial to note that any Killing field $\boldsymbol{w}_{\boldsymbol{f}}$ is determined by a *single* parameter vector $\boldsymbol{f} \in \mathbb{R}^6$, which we choose from a *fixed* position x in the frame field $x \mapsto \boldsymbol{f}(x)$. That is, given a position x we determine $\boldsymbol{f} := \boldsymbol{f}(x)$, and then $\boldsymbol{f}$ determines the *entire* field $\boldsymbol{w}_{\boldsymbol{f}}$. To make this more explicit, we refer to positions in the domain U as either points $x \in U$ or $y \in U$, depending on context, to differentiate between the field of frame parameters $x \mapsto \boldsymbol{f}(x)$ and a particular Killing field $y \mapsto \boldsymbol{w}_{\boldsymbol{f}}(y)$ corresponding to just one 6D frame parameter vector $\boldsymbol{f}$, respectively. Nevertheless, positions x and y refer to points in the same domain U.

In this way, because in our setup each vector $\boldsymbol{f}(x)$ at a single position $x \in U$ defines a Killing field $y \mapsto \boldsymbol{w}_{\boldsymbol{f}}(y)$ over the whole domain U, instead of being limited to just a single reference frame, as one Killing field $\boldsymbol{w}$, we thus define a whole *field* of reference frames.

For the direct evaluation of Killing vector fields $\boldsymbol{w}_{\boldsymbol{f}}$ using Eq. 7 with parameters $\boldsymbol{f}$, and the basis Killing fields $\boldsymbol{e}_i$ given in Eq. 9, at each sample point $y_i \in U$ we define the corresponding 2×3 matrix

$$\mathbf{W}(y_i) = \begin{pmatrix} 1 & 0 & -(\hat{\mathrm{y}}(y_i) - \hat{\mathrm{y}}_m) \\ 0 & 1 & (\hat{\mathrm{x}}(y_i) - \hat{\mathrm{x}}_m) \end{pmatrix}. \tag{11}$$

The points y_i, $i \in [1, N]$, are the spatial positions of N grid points $y_i = x_i \in U$, on which the input vector field $\boldsymbol{v}$ is given (i.e., sampled).

Using the matrices $\mathbf{W}(y_i)$, for any 6D parameter vector $\boldsymbol{f}$, we compute the corresponding Killing field and its time partial as

$$\boldsymbol{w}_{\boldsymbol{f}}(y_i) = \begin{pmatrix} \mathbf{W}(y_i) \mid \mathbf{0} \end{pmatrix} \boldsymbol{f}, \tag{12}$$

$$\frac{\partial \boldsymbol{w}_{\boldsymbol{f}}}{\partial t}(y_i) = \begin{pmatrix} \mathbf{0} \mid \mathbf{W}(y_i) \end{pmatrix} \boldsymbol{f}. \tag{13}$$

Like the matrices $\mathbf{W}$, the zero matrices $\mathbf{0}$ each are of size 2×3.

3.2 Moving Least Squares Approximation

Using moving least squares, we estimate the reference frame parameters $\boldsymbol{f}(x)$ at any position $x \in U$ using a local weighted least squares approximation, where the weighting is determined by a function $\theta(d)$.

However, because the function that we want to approximate will be the *observed time derivative* [20, 60] of an input flow field $\boldsymbol{v}$ with respect to the frame field $\boldsymbol{f}(x)$ (see Eqs. 17 and 18), and to make our notation match the intended semantics, we will now denote the input function values that we want to approximate by $\mathscr{D}_i$, instead of by f_i as in Eq. 4, and the approximating function by $\mathscr{D}(x)$, instead of by $f(x)$.

For any given position $x \in U$, which does not have to be a grid point, we determine the corresponding unknown 6D parameter vector $\boldsymbol{f}(x) \in \mathbb{R}^6$, such that the approximating function $\mathscr{D}(x)$ determined by $\boldsymbol{f}(x)$ minimizes a weighted squared approximation error. Using an MLS formulation with a vector-valued function $\mathscr{D}(x)$ [39], we can write

$$\boldsymbol{f}(x) = \operatorname{argmin}_{\boldsymbol{f}'} \sum_{i \in I(x)} \left\| \mathscr{D}_{\boldsymbol{f}'}(x_i) - \mathscr{D}_i \right\|^2 \theta(\|x - x_i\|). \tag{14}$$

The vector-valued input function is only known by sample values $\mathscr{D}_i$ at discrete sample positions x_i. The aim is to determine a *smooth* function $\mathscr{D}(x)$ that approximates the function values $\mathscr{D}_i$, with the squared approximation error measured at the corresponding points x_i. The function $\mathscr{D}(x)$ is determined by the parameters that are encoded in the smooth parameter field $\boldsymbol{f}(x)$. We search for the unknown parameters $\boldsymbol{f}$, where for every point x the corresponding parameters $\boldsymbol{f}(x)$ are determined by solving a local weighted least squares approximation problem. Solving this problem for all points $x \in U$ gives the smooth parameter field $\boldsymbol{f}(x)$ and the corresponding smooth function $\mathscr{D}(x)$.

The function $\mathscr{D}(x)$ and the discrete function values $\mathscr{D}_i$ at positions x_i will be defined below such that the least squares approximation problem of Eq. 14 will determine an *optimal reference frame field* of parameters $\boldsymbol{f}(x)$ that is locally adapted to features of the input flow field $\boldsymbol{v}$.

The weights in Eq. 14 are given by the distance-dependent weighting function $\theta(d)$, where $d = \|x - x_i\|$ is the distance between the points x and x_i. Typical weighting functions in moving least squares approximation, which we also use in this work, are the Gaussian

$$\theta(d) = e^{-\lambda d^2}, \tag{15}$$

and the exponential

$$\theta(d) = e^{-\lambda d}. \tag{16}$$

In these weighting functions, $\lambda \in \mathbb{R}$ is a user-defined scaling parameter. In the supplemental material, we also evaluate Wendland's compactly supported functions $\phi_{3,0}$, $\phi_{3,1}$, and $\phi_{3,2}$ [56], which yield results similar to those of the Gaussian function.

To locally adapt the reference frame field to features of the flow field $\boldsymbol{v}$, we modify the distance function between points x and x_i according to a scalar *guidance field*, as described below, yielding our final moving least squares approximation in Eq. 22.

3.2.1 Fitting reference frame fields

In order to compute a moving least squares approximation using the formulation of Eq. 14, we have to define the input function values $\mathscr{D}_i$ given at discrete sample positions x_i, and how the corresponding approximating smooth function $\mathscr{D}(x)$ is determined by the parameters $\boldsymbol{f} \in \mathbb{R}^6$. To determine a field of unknown reference frame parameters $\boldsymbol{f}(x)$, we compute the minimizer in Eq. 14 with the definitions

$$\mathscr{D}_{\boldsymbol{f}}(x_i) := \frac{\partial \boldsymbol{v}}{\partial t} - \frac{\partial \boldsymbol{w}_{\boldsymbol{f}}}{\partial t} + \nabla \boldsymbol{v}(\boldsymbol{w}_{\boldsymbol{f}}) - \nabla \boldsymbol{w}_{\boldsymbol{f}}(\boldsymbol{v}), \tag{17}$$

$$\mathscr{D}_i := (0,0)^T. \tag{18}$$

On the right-hand side of Eq. 17, the dependence of all terms on the point x_i is implicit for brevity. Likewise, all terms involving $\boldsymbol{w}$ are determined by the parameter vector $\boldsymbol{f}$, as given by Eqs. 7 and 8.

The term $\nabla \boldsymbol{v}(\boldsymbol{w})$ is the directional derivative of $\boldsymbol{v}$ in direction $\boldsymbol{w}$, i.e., $\nabla \boldsymbol{v}(\boldsymbol{w}) = \nabla_{\boldsymbol{w}} \boldsymbol{v}$. In fluid mechanics, this is sometimes also written as directional derivative operator $(\boldsymbol{w} \cdot \nabla)\,\boldsymbol{v}$. Likewise, $\nabla \boldsymbol{w}(\boldsymbol{v}) = \nabla_{\boldsymbol{v}} \boldsymbol{w}$ is the directional derivative of $\boldsymbol{w}$ in direction $\boldsymbol{v}$, or $(\boldsymbol{v} \cdot \nabla)\,\boldsymbol{w}$.

Eq. 17 specifies the vector corresponding to the *observed time derivative* defined in prior work, e.g., [20, 21, 60], using the specific Lie derivative $(\partial/\partial t + \mathscr{L}_{\boldsymbol{w}})(\boldsymbol{v} - \boldsymbol{w})$, evaluated at the point x_i. The observed time derivative is a vector field that gives the instantaneous rate of change of the velocity field $\boldsymbol{v}$ relative to the reference frame motion given by the velocity field $\boldsymbol{w}$, which in our context is determined by the parameter vector $\boldsymbol{f}$. The goal of minimizing this expression is to determine a reference frame $\boldsymbol{w}$ that perceives an unsteady input flow field $\boldsymbol{v}$ *as steady as possible*. We also note that this expression is objective [20]. Correspondingly, all function values $\mathscr{D}_i$ in Eq. 18 are simply defined to be the zero vector at all x_i, because we target a vanishing time derivative, i.e., a steady observed velocity field, everywhere.

This leads to a somewhat "non-standard" moving least squares approximation problem, because we are in fact approximating an everywhere-vanishing function. However, this enables us to use the standard moving least squares machinery in our context, and it gives us the desired smooth reference frame field $\boldsymbol{f}(x)$ over the whole domain U.

3.2.2 Using guidance fields

We conceptually embed the regular grid of the input vector field $\boldsymbol{v}$, with grid points x_i, in a manifold on which the regularly spaced grid points become irregularly spaced points $\tilde{x}_i$. To determine the irregular spacing of points, we use a scalar field, such as FTLE, as a *guidance field*

$$g\colon U \to \mathbb{R}. \tag{19}$$

We use the field g to conceptually define a *height field*, as a two-dimensional manifold (i.e., a surface) embedded in $\mathbb{R}^3$. We define this manifold as the graph of the function $g(x)$, which is given by the set

$$G = \{(x, g(x)) | x \in U\} \subset \mathbb{R}^2 \times \mathbb{R}. \tag{20}$$

Over the 2-manifold G, with points $\tilde{x} \in \mathbb{R}^2 \times \mathbb{R}$ on G denoted by $\tilde{x} := (x, g(x))$, we will compute distances

$$d_g(x, x_i) := distance(\tilde{x}, \tilde{x}_i). \tag{21}$$

That is, the distance function d_g depends on the guidance field g and the corresponding manifold G. The *distance* function on the right-hand side above is discussed in Sec. 3.2.4. Using the distances d_g determined by the guidance field g, and inserting $\mathscr{D}_i = 0$ (Eq. 18), we finally obtain the complete moving least squares approximation that we use as

$$\boldsymbol{f}(x) = \operatorname{argmin}_{\boldsymbol{f}'} \sum_{i \in I(x)} \left\| \mathscr{D}_{\boldsymbol{f}'}(x_i) \right\|^2 \theta\left(d_g(x, x_i)\right). \tag{22}$$

The index set $I(x)$ can be chosen such that, for all $i \in I(x)$, either $d_g(x, x_i) < r$, with some distance threshold $r \in \mathbb{R}$, or alternatively such that $\theta\left(d_g(x, x_i)\right) > \varepsilon$, for some small weight threshold $\varepsilon \in \mathbb{R}$ below which the influence of a point is considered to be negligible. In our implementation, we use the latter approach, i.e., we look at weight values $\theta\left(d_g(x, x_i)\right)$ and set a small threshold ε. We choose this approach for convenience, and note that alternatively specifying a distance threshold r is completely equivalent, since all our weighting functions θ are invertible. Thus, for any given weight threshold ε, the corresponding unique distance $r = \theta^{-1}(\varepsilon)$ can be determined directly, and vice versa.

To facilitate efficient GPU implementation, we use a conservative approach to choose $I(x)$ to correspond to a square region of grid points of the input grid around the point x. Conservative here means that the region should not be too small (i.e., it should not miss x_i with $\theta\left(d_g(x, x_i)\right) > \varepsilon$), but it can be slightly too large such that it might contain some x_i with $\theta\left(d_g(x, x_i)\right) \leq \varepsilon$. For a given data set, we choose the same constant region size for all points $x \in U$, and verify in a pre-process that no weight values greater than ε are missed. See below.

3.2.3 Explicit local weighted least squares computation

For any given point y_i, building on Eq. 11, and Eqs. 12 and 13, we define the following 2×6 matrix $\mathbf{A}$, and 2×1 right-hand side $\boldsymbol{b}$, respectively,

$$\begin{aligned} \mathbf{A}(y_i) &= \left(\left(\mathbf{0} \mid -\mathbf{W}(y_i) \right) + \left(\nabla \boldsymbol{v} \cdot \mathbf{W}(y_i) \mid \mathbf{0} \right) - \left(\begin{array}{cc|c} 0 & 0 & -v_2 \\ 0 & 0 & v_1 \end{array} \right) \right), \\ \boldsymbol{b}(y_i) &= -\frac{\partial \boldsymbol{v}}{\partial t}. \end{aligned} \tag{23}$$

The dependence of $\boldsymbol{v}$ on the point y_i, as for $\mathbf{W}$, is implicit for brevity, and the vector $\boldsymbol{v} = (v_1, v_2)^T$. The matrix $\mathbf{A}$ results from Eqs. 12 and 13, and using Eq. 10 for the term $\nabla \boldsymbol{w}_{\boldsymbol{f}}(\boldsymbol{v})$. We can now compute Eq. 17 as

$$\mathscr{D}_{\boldsymbol{f}}(y_i) = \mathbf{A}(y_i)\, \boldsymbol{f} - \boldsymbol{b}(y_i). \tag{24}$$

We now define (but never explicitly construct) a $2N \times 6$ matrix $\bar{\mathbf{A}}$ vertically concatenating all 2×6 matrices $\mathbf{A}(y_i)$ for N points y_i, with $N = |I(x)|$ the cardinality of the index set $I(x)$, and similarly a $2N \times 1$ vector $\bar{\boldsymbol{b}}$. The *normal equations* of the corresponding weighted least squares problem determining the minimizer $\boldsymbol{f} = \boldsymbol{f}(x)$, x fixed, are then

$$\begin{aligned} \bar{\mathbf{A}}^T \boldsymbol{\Theta}\, \bar{\mathbf{A}}\, \boldsymbol{f} &= \bar{\mathbf{A}}^T \boldsymbol{\Theta}\, \bar{\boldsymbol{b}}, \\ \boldsymbol{f} &= \left(\bar{\mathbf{A}}^T \boldsymbol{\Theta}\, \bar{\mathbf{A}} \right)^{-1} \mathbf{A}^T \boldsymbol{\Theta}\, \bar{\boldsymbol{b}}. \end{aligned} \tag{25}$$

The (not explicitly constructed) $2N \times 2N$ weight matrix $\boldsymbol{\Theta}$ is

$$\boldsymbol{\Theta} = diag\left(\theta\left(d_g(x, x_1)\right), \ldots, \theta\left(d_g(x, x_N)\right) \right). \tag{26}$$

We directly compute the 6×6 matrix $\bar{\mathbf{A}}^T \boldsymbol{\Theta} \bar{\mathbf{A}}$, at any point x, as ($x_i = y_i$)

$$\left(\bar{\mathbf{A}}^T \boldsymbol{\Theta} \bar{\mathbf{A}} \right)(x) = \sum_{i \in I(x)} \theta\left(d_g(x, x_i)\right) \mathbf{A}^T(y_i)\, \mathbf{A}(y_i). \tag{27}$$

Similarly, we get the 6×1 right-hand side $\bar{\mathbf{A}}^T \boldsymbol{\Theta}\, \bar{\boldsymbol{b}}$, at the point x, as

$$\left(\bar{\mathbf{A}}^T \boldsymbol{\Theta}\, \bar{\boldsymbol{b}} \right)(x) = \sum_{i \in I(x)} \theta\left(d_g(x, x_i)\right) \mathbf{A}^T(y_i)\, \boldsymbol{b}(y_i). \tag{28}$$

The solution for the minimizer $\boldsymbol{f}(x)$ at any desired point $x \in U$ is thus computed by solving the 6×6 square linear system given by Eq. 25 for the point x, with the 6×6 system matrix $(\bar{\mathbf{A}}^T \boldsymbol{\Theta} \bar{\mathbf{A}})(x)$ computed via Eq. 27, and the 6×1 right-hand side $(\bar{\mathbf{A}}^T \boldsymbol{\Theta}\, \bar{\boldsymbol{b}})(x)$ computed via Eq. 28.

The 6×6 system matrix $\bar{\mathbf{A}}^T \boldsymbol{\Theta} \bar{\mathbf{A}}$ is always guaranteed to be positive semi-definite, and if it is non-singular, it is positive definite. To ensure non-singularity and numerical stability, we solve our system with Tikhonov regularization [52]. In our implementation, we solve the regularized linear system via the standard QR factorization of $\bar{\mathbf{A}}^T \boldsymbol{\Theta} \bar{\mathbf{A}} + \lambda_T \mathbf{I}$, where λ_T is a regularization weight and $\mathbf{I}$ is the identity matrix.

Computing the minimizer $\boldsymbol{f}(x)$ for all $x \in U$ gives the smooth reference frame field $x \mapsto \boldsymbol{f}(x)$. In our implementation, we compute the minimizer at the spatial locations x_i of the grid points of the regular grid on which the input vector field $\boldsymbol{v}$ was sampled, and store the corresponding 6D parameter vector $\boldsymbol{f}(x_i)$ (Eq. 6) per grid point x_i.

3.2.4 Distance function

In order to define the *distance* function in Eq. 21, we will use curves $s \mapsto \tilde{\gamma}(s)$ on the height field manifold G, i.e., we define curves

$$\begin{aligned} \tilde{\gamma}\colon I &\to G, \\ s &\mapsto \tilde{\gamma}(s). \end{aligned} \tag{29}$$

The interval I is a parameter interval $I \subset \mathbb{R}$, such as $I = [0, 1]$. We define the curves $\tilde{\gamma}$ on G by "lifting" curves $s \mapsto \gamma(s)$ in the domain U (i.e., any γ is a curve in flat $\mathbb{R}^2$) to the (curved) height field manifold G, according to the guidance field g. That is, we define the curves

$$\tilde{\gamma}(s) := \left(\gamma(s), g(\gamma(s)) \right) \in G \subset \mathbb{R}^2 \times \mathbb{R}. \tag{30}$$

The corresponding curve in $U \subset \mathbb{R}^2$ is given by

$$\begin{aligned} \gamma\colon I &\to U, \\ s &\mapsto \gamma(s). \end{aligned} \tag{31}$$

One option to define distances on the height field manifold G would be to use geodesic distances on the manifold G. However, for simplicity, we define the curves γ as straight lines in $U \subset \mathbb{R}^2$, connecting the points x and x_i. We define the curve (straight line)

$$\gamma(s) := x + \frac{s - s_a}{s_b - s_a} \cdot (x_i - x), \quad \text{with} \quad s \in I, \tag{32}$$

with the interval $I = [s_a, s_b]$, for $\gamma\colon I \to U$, given by

$$\begin{aligned} s_a &:= 0, \\ s_b &:= \|x_i - x\|. \end{aligned} \tag{33}$$

That is, the parameterization of γ uses the Euclidean distance in $\mathbb{R}^2$, trivially giving an arc length parameterization of the straight line γ.

From this basic setup, we can now define a distance measure based on the arc length of the curve $\tilde{\gamma}$ *on the height field manifold* G. That is, we define the distance $d_g(x, x_i)$ to be the arc length of the curve $s \to \tilde{\gamma}(s)$ on G, from the point $\tilde{x} = (x, g(x))$ to the point $\tilde{x}_i = (x_i, g(x_i))$, i.e.,

$$d_g(x, x_i) := \int_{s_a}^{s_b} \left\| \frac{\mathrm{d}\tilde{\gamma}(s)}{\mathrm{d}s} \right\| \mathrm{d}s. \tag{34}$$

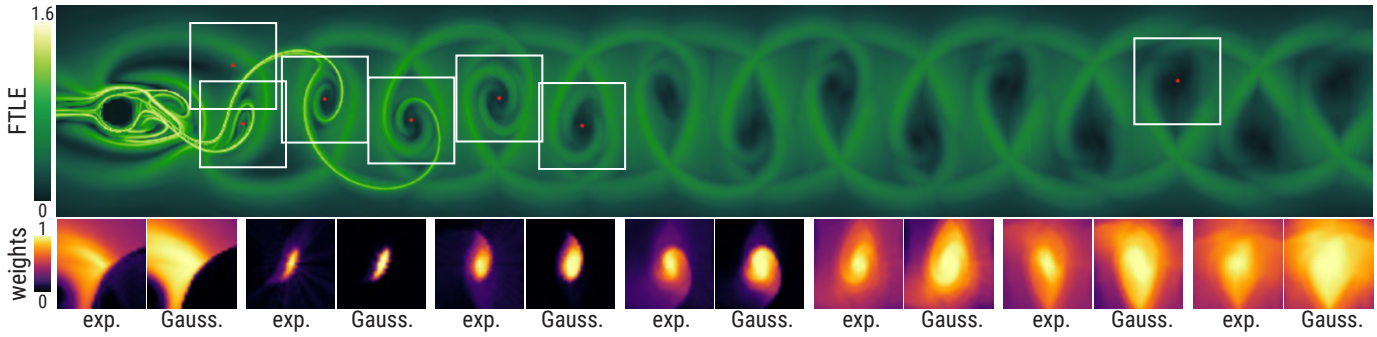

Fig. 4: **Kármán vortex street**. (Top) FTLE field at time $t = 10.8$. (Bottom) Inset images, presented in pairs of exponential (Eq. 16) and Gaussian (Eq. 15) weighting functions, color-code the weights $\theta(d)$ of the spatial neighborhood surrounding the red points. Gaussian weighting produces localized weights, while exponential weighting produces large regions of nonzero weights. From left to right, the first pair of insets corresponds to the top-left point, the second corresponds to the bottom-left point, and the remaining correspond to points moving towards the right. As vortices move away from the obstacle and fade, their separatrices also fade, and the region they enclose expands. The neighborhood weighting adapts accordingly.

This is illustrated with the red line and arc in Fig. 3. A simple way to compute the arc length in this setting is to use Cartesian coordinates $(\hat{x}, \hat{y}, \hat{z})$, where $\hat{x}(\gamma(s))$ and $\hat{y}(\gamma(s))$ are the coordinates of the point $\gamma(s)$ in the domain $U \subset \mathbb{R}^2$, and the coordinate $\hat{z}$ is defined by

$$\hat{z}(\gamma(s)) := g(\gamma(s)). \tag{35}$$

In order to compute Eq. 34, we therefore need to compute

$$\left\| \frac{\mathrm{d}\tilde{\gamma}(s)}{\mathrm{d}s} \right\| = \sqrt{\left(\frac{\mathrm{d}\hat{x}}{\mathrm{d}s}\right)^2 + \left(\frac{\mathrm{d}\hat{y}}{\mathrm{d}s}\right)^2 + \left(\frac{\mathrm{d}\hat{z}}{\mathrm{d}s}\right)^2}. \tag{36}$$

We sample the curve from $\tilde{x}$ to $\tilde{x}_i$ with short straight line segments with parameter length Δs, integrating Eq. 34 numerically. If the segments are small enough, at any point with parameter s we can approximate

$$\left(\frac{\mathrm{d}\hat{z}}{\mathrm{d}s}\right)\Bigg|_s \approx \frac{g(\gamma(s+\Delta s)) - g(\gamma(s))}{\Delta s}. \tag{37}$$

Moreover, the curve parameterization of Eq. 33 means that $(\mathrm{d}\hat{x}/\mathrm{d}s)^2 + (\mathrm{d}\hat{y}/\mathrm{d}s)^2 = 1$, and we can approximate the integrand in Eq. 34 by

$$\left\| \frac{\mathrm{d}\tilde{\gamma}(s)}{\mathrm{d}s} \right\| \mathrm{d}s \approx \sqrt{1 + \left(\frac{g(\gamma(s+\Delta s)) - g(\gamma(s))}{\Delta s}\right)^2} \cdot \Delta s. \tag{38}$$

3.2.5 FTLE as guidance field

To constrain the neighborhood of the reference frame optimization to regions with coherent fluid motion, we use the maximum of the forward and backward finite-time Lyapunov exponent as guidance field $g(x)$ in the distance computation, cf. Eq. 1;

$$g(x) = \max\left(0, \sigma_{t_0}^{T}(x), \sigma_{t_0}^{-T}(x)\right). \tag{39}$$

To resolve small-scale structures, we calculate FTLE fields with three times the resolution of the input vector field. The size dim_{FTLE} of each dimension is computed as $3 \cdot dim_{input} - 2$. The necessary flow map gradient $\mathbf{F}$ is computed numerically by finite differences [22] from flow maps released at the higher resolution grid. For each sample in the FTLE field, we release four tracer particles around the sample (up, down, left, right), at distances (domain size)$/(dim_{FTLE} - 1)$, dimensions and spacings are independent for each dimension. For the pathlines, the integration is computed using $\Delta t_{FTLE} = \Delta t/4$, where Δt is the input field temporal sampling rate. FTLE is dependent on the integration duration T. The effect of this parameter on the result is studied below. For a given time and position, the weight or contribution of each neighboring sample to the local frame optimization is computed based on the induced distance, which can be large for spatially close points in the input domain, if they are positioned on opposite sides of an FTLE ridge. In this way, the reference frame optimization will locally adjust to the regions defined by the Lagrangian coherent structures present in the FTLE field, as illustrated in Fig. 4 for the Kármán vortex street.

4 Results

After introducing the data sets, we first formally define our quantitative quality metrics. Next, we perform a parameter study of the FTLE integration time and provide details on the calculation of the FTLE fields. Then, we discuss the selection of the parameter λ for each dataset. Afterwards, we discuss the decreased observed time derivative and small-scale vortices detected using our pipeline. Finally, we show a runtime comparison between our method and the baseline.

Data sets. We evaluate our method on five numerically simulated turbulent vector fields computed using a stream function vorticity solver in the open-source framework Basilisk [41]. In the following, we refer to these flows as Turbulence1 (least turbulent) to Turbulence5 (most turbulent). The first four flows have dimensions $256 \times 256 \times 400$ and are defined in the space-time domain $[0,1]^2 \times [0,0.2]$. The last flow has size $256 \times 256 \times 321$, and is defined in $[0,1]^2 \times [0,0.1605]$. For reference, we then also apply the approach to a well-known Cylinder flow [13], containing a von Kármán vortex street at Reynolds number 160. The flow was simulated with Gerris [42], has a resolution of $640 \times 80 \times 1501$, and is defined in the space-time domain $[-0.5,7.5] \times [-0.5,0.5] \times [0,15]$. For validation, we process all datasets with the local frame optimization method by Günther et al. [13], which we denote 'Baseline'. We extract vortex corelines in the optimal frame by tracking the corresponding critical points moving over time, requiring complex eigenvalues in the Jacobian at the critical point to exhibit rotational behavior [16]. For all results, we use the Gaussian weighting function (Eq. 15). It is smooth at $d = 0$ and its faster decay after $d > 1$ provides better locality than the exponential function, as illustrated in Fig. 4. For all experiments, we use $\lambda_T = 10$ for Tikhonov regularization when solving the linear system of Eq. 25.

Quality criteria. For quantitative comparisons of the extracted vortex corelines, we compute a number of quality criteria. Let C be the set of all vortex corelines. Each coreline is denoted by $\mathbf{c} \in C$, and is represented by samples at discrete positions $c_i \in \mathbb{R}^2$ along the coreline, where c_i denotes the coreline position at time t_i. First, we calculate the *duration* of all discrete vortex corelines $\mathbf{c} \in C$, as the number of timesteps they span, which is later referred to as m_{tac}.

The alignment of vortex corelines and pathlines is assessed per segment by releasing a pathline at the start of the segment and measuring the distance to the end of the segment after advection, giving a *mean*

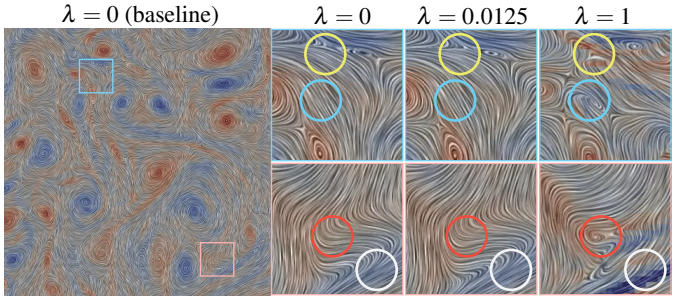

Fig. 5: LIC in the optimal frame with color-coded vorticity, TURBULENCE1 dataset. Small-scale vortices outside the current turbulence spectrum are observed alongside artifacts in the color-coded vorticity for $\lambda = 1$. For $\lambda = 0.0125$, the results are consistent with the baseline.

segment error when averaged over all segments:

$$m_{\text{mse}} = \frac{1}{|C|} \sum_{\mathbf{c} \in C} \frac{1}{|\mathbf{c}|} \sum_{(c_i, c_{i+1}) \in \mathbf{c}} \| \phi_{t_i}^{t_{i+1}-t_i}(c_i) - c_{i+1} \|. \tag{40}$$

Here, as in all our FTLE computations, ϕ denotes the flow map. The per-segment error is later color-coded on the visualized vortex corelines. The drift of corelines from pathlines is determined by an *endpoint error*, as the difference between the endpoint of the coreline and the final position (endpoint) of the pathline released at the initial position of the coreline:

$$m_{\text{epe}} = \frac{1}{|C|} \sum_{\mathbf{c} \in C} \| \phi_{t_0}^{t_n - t_0}(c_0) - c_n \|. \tag{41}$$

We also include the total number of detected corelines. For non-turbulent flows, this metric should be similar between the baseline and our method. However, for turbulent flows, our method should detect a larger number of corelines corresponding to the previously undetected vortices at smaller scales, while maintaining low average error across the corelines.

Finally, we include the integral over the whole domain of the magnitude of the observed time derivative, given by the right-hand side of Eq. 17. In this section, we will denote the observed time derivative by $\boldsymbol{v}_t^*$, and the magnitude of the observed time derivative integrated over the whole domain by $\| \boldsymbol{v}_t^* \|_{total}$. This is an important quantity to evaluate, since the frame optimization aims to observe flow fields as steady as possible, i.e., to minimize $\| \boldsymbol{v}_t^* \|_{total}$.

4.1 FTLE Guidance Field Computation

For each timestep, we compute a 2D FTLE field, yielding a time-dependent 2D scalar field. We define the integration time as $T_{FTLE} = k \cdot \Delta t_{FTLE}$, where k is a given number of discrete timesteps. The initial time t_0 is set to t, the time of the current timestep. Repelling FTLEs are computed with $T = T_{FTLE}$, and attracting FTLEs with $T = -T_{FTLE}$. To understand the effect of k in the guidance fields, we computed a series of FTLE fields for TURBULENCE1 with $k = 1, 5, 10, 20, 50, 80, 100, 200$. FTLE images for multiple values of k are included in the supplementary material. Noticeable differences were found between $k = 10, 50, 100, 200$. For lower values of k, the separatrices are blurred, and they become sharper as k increases. Furthermore, for fixed $k = 80$, we computed FTLE fields across the turbulence datasets. As turbulence increases, we get sharper FTLE features.

We use $k = 80$ for all datasets. Larger values will improve the quality of the guidance fields at lower levels of turbulence. However, the baseline method performs well in this scale, and the runtime increases with k. Hence, we decided to use $k = 80$ for all our experiments, which provided moderate coreline improvements for laminar flows, and significant improvements for turbulent flows. For all datasets, we compute and store $\sigma_{t_0}^{T}(x)$ and $\sigma_{t_0}^{-T}(x)$ before optimization for all available timesteps. For the first k timesteps, it is not possible to compute the required attracting field, and similarly for the repelling field in the final k timesteps.

4.2 Parameter Study

The weighting functions in the moving least squares approximation in Eqs. 15, 16 contain a user-specified parameter λ. In Table 1, we varied λ for the TURBULENCE1 dataset, observing its impact on the quantitative metrics for our approach, compared with the baseline. As λ increases, the number of corelines increases and the average length decreases significantly. The average m_{epe} (Eq. 41) decreases, while the average m_{mse} (Eq. 40) increases. Hence, higher values of λ produce multiple short, high error corelines. This is illustrated in Fig. 5, where false positives are introduced for $\lambda = 1$. Regarding the drop in average m_{epe}, it is likely due to the additional shorter corelines. As λ decreases, the overall metrics improve until $\lambda = 0.0125$, which is the value that we used for the TURBULENCE1 dataset, as it resulted in smaller errors and the longest vortex corelines. In Fig. 5, we also emphasize that $\lambda = 0.0125$ is consistent with the baseline, which is equivalent to $\lambda = 0$ since we get $\theta(d) = 1$ independently of d. The results are virtually identical for large-scale features, with minor differences at the smallest scales. Additional comparison images for large coherent regions are presented in the supplementary material.

We performed similar experiments for the other datasets and found similar behavior while varying λ. We selected $\lambda = 0.0125$ for TURBULENCE2, and $\lambda = 0.00625$ for datasets TURBULENCE3 to TURBULENCE5. For the CYLINDER dataset, we chose $\lambda = 0.02$. Tables with metric comparisons are found in the supplementary material for the other datasets.

4.3 Kármán Vortex Street

We optimize this well-known vector field with our method and the baseline method as validation. With $\lambda = 0.02$, we obtain almost identical results to the baseline. The coreline metrics are included in the supplementary material.

Away from the obstacle, the corelines and LIC images are identical. Only the corelines directly on the right of the obstacle are slightly different, which is expected since this is the only region with relatively small-scale features corresponding to newly born vortices. In the rest of the domain, features are relatively large-scale and move coherently in a straight path towards the right. Near the obstacle, our corelines are longer by one segment and have marginally higher error, as seen in Fig. 6. The leftmost long coreline in ours is slightly longer towards earlier timesteps. For the shorter coreline at the top left, ours is slightly longer towards $-t$, and the color-coded segment error reveals a (numerically marginal) higher error. LIC and $\| \boldsymbol{v}_t^* \|$ for the leftmost part of the domain are shown in the supplementary material.

4.4 Decreasing Observed Time Derivative

Frame optimization seeks to observe vector fields in an as-steady-as-possible form, which relates to minimizing the magnitude of the observed time derivative, given by the right-hand side of Eq. 17. In

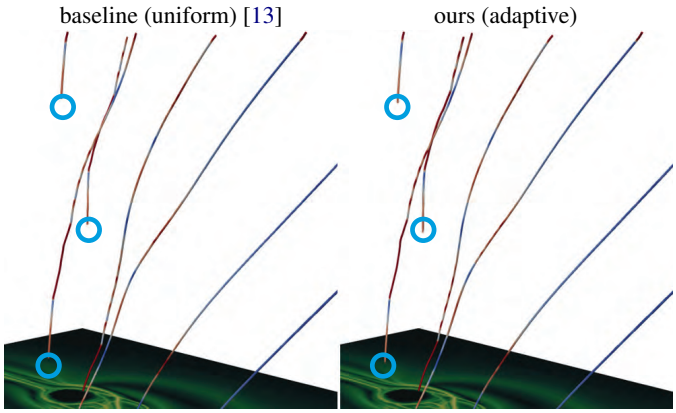

Fig. 6: 2D space-time view of different corelines. Time is depicted along the vertical axis. Our results are consistent with the baseline, with some slightly longer corelines.

Table 1: **Parameter study** for the TURBULENCE1 dataset to determine a suitable choice for λ in Eq. 15. The reported values are averages over all corelines for each setting. m_{tac} is the duration of corelines in timesteps, m_{mse} is Eq. 40, m_{epe} is Eq. 41, $|C|$ is the number of corelines, and $\|\boldsymbol{v}_t^*\|_{total}$ is the magnitude of the observed time derivative integrated over the whole domain.

Quality Criteria	$\lambda = 1$	$\lambda = 0.5$	$\lambda = 0.25$	$\lambda = 0.025$	$\lambda = \mathbf{0.0125}$	$\lambda = 0.00625$	Baseline
m_{tac}	58.72	60.97	70.96	89.69	93.85	88.67	90.18
m_{mse}	2.39×10^{-4}	2.31×10^{-4}	2.03×10^{-4}	1.74×10^{-4}	1.73×10^{-4}	1.98×10^{-4}	1.83×10^{-4}
m_{epe}	2.91×10^{-3}	2.77×10^{-3}	3.14×10^{-3}	4.19×10^{-3}	4.61×10^{-3}	4.45×10^{-3}	4.378×10^{-3}
$\|C\|$	342	345	304	212	200	209	204
$\|\boldsymbol{v}_t^*\|_{total}$	2.33×10^7	2.53×10^7	2.81×10^7	3.76×10^7	3.93×10^7	4.04×10^7	4.18×10^7

Fig. 7, we present crops of the FTLE and the magnitude of the observed time derivative $\|\boldsymbol{v}_t^*\|$ of the objective fields obtained with the baseline and our method, for timestep 80 of the TURBULENCE5 dataset. Our method produces a smaller $\|\boldsymbol{v}_t^*\|$ in the entire domain. Moreover, significant differences are found in multiple regions corresponding to FTLE valleys surrounded by ridges. As marked by the circles, the magnitude of the observed time derivative obtained with the baseline presents strong spikes, which are not present in our results.

For this highly turbulent dataset, there is a significant difference in the number of detected corelines $|C|$ among the considered methods: 1886 for the baseline, and 2389 for our method. All the remaining metrics are also improved with our technique. Regarding the impact of our method on the minimization of the magnitude of the observed time derivative, its integral over the whole (space-time) domain is 2.11×10^8 with the baseline, and it is 1.48×10^8 with our method, roughly 30% less. In contrast, for non-turbulent flows, there are large valleys in the FTLE field where the optimization results obtained with the baseline and our technique are similar, as seen in Fig. 5. Please refer to the supplementary material for images including FTLE, LIC, and $\|\boldsymbol{v}_t^*\|$ for TURBULENCE1.

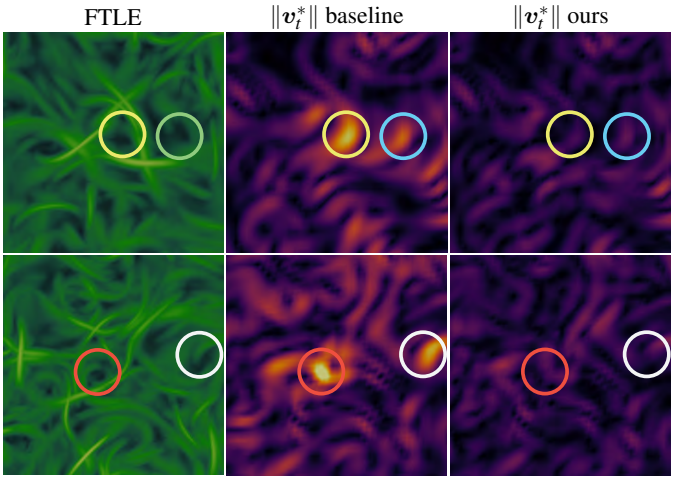

Fig. 7: Comparison of the magnitude of the observed time derivative, $\|\boldsymbol{v}_t^*\|$, of the baseline (center) and our method (right) for the TURBULENCE5 dataset. Overall, the $\|\boldsymbol{v}_t^*\|$ obtained with our method is smaller. Moreover, significant differences are found in several regions corresponding to FTLE (left) valleys, where the baseline presents spikes in $\|\boldsymbol{v}_t^*\|$.

4.5 Detection of Small-Scale Vortices

Our method can detect small-scale vortices since we weight the samples in the neighborhood adaptively based on the FTLE field. Fig. 1 shows the FTLE field and the result of the reference frame optimizations of prior work [13] and our adaptive method for our most turbulent flow, TURBULENCE5. It is evident that the FTLE field, comprising both forward and backward FTLE, segments the flow into a very high number of regions, some of which enclose small-scale vortices. The close-ups below the LIC images show several smaller vortices that have been successfully detected by our method, but not by prior work. The color-coding of vorticity reveals where vortices are expected to appear. In prior work, regions of vorticity extrema were observed in LIC images; however, LIC does not explicitly depict vortical motion. In contrast, our adaptive method generated LIC images with circling streamlines in regions with vorticity extrema and bounded by FTLE ridges.

4.6 Corelines in the Turbulence Datasets

We showed in previous sections that the observed time derivative, which is the quantity minimized by the optimization, decreases when using our adaptive neighborhoods. To evaluate the effect of our method on coreline extraction, we compare corelines extracted using prior work [13] and our adaptive method for datasets TURBULENCE1, TURBULENCE3, and TURBULENCE5 in Fig. 8. Similar comparisons are included for datasets TURBULENCE2 and TURBULENCE4 in the supplementary document. Moreover, these comparisons are further illustrated in the supplementary video. For both methods, a neighborhood of 31×31 grid points was used. The corelines have been extracted by a root-finding solver per time slice [16] and connected over time to form connected curves. To depict coreline quality, the mean segment error m_{mse} is color-coded on the lines. We can observe that the differences in the corelines increase as the flow turbulence increases. This is not surprising, since the uniformly-weighted reference frame optimization [13] places the neighborhood window over multiple vortices. If those smaller vortices travel in different directions, then it is not possible to find a single observer following all vortices in the neighborhood. Instead, a compromise is found that cannot make the flow steady for either of the smaller vortices. Consequently, prior methods fail to capture small-scale structures. In addition to detecting additional vortices, we observed that our method allows the extraction of longer and smoother vortex corelines.

4.7 Runtime Comparison

Adjusting the neighborhood in the reference frame optimization according to the transport barriers comes at a performance cost, due to the computation of the FTLE and guidance field. Table 2 compares the runtime between the baseline [13] and our method. All measurements were taken on a workstation with an Intel Xeon CPU E5-2687W (3 GHz) and an Nvidia GeForce RTX3090. We implemented the reference frame optimization for both our method and the baseline [13] on the CPU with parallelization. The baseline method optionally uses summed-area tables (SAT) for the sum of the local linear systems. This

Table 2: Comparison of the **computation times** between the baseline [13] and our method. For the baseline, we show the performance with and without the use of summed area tables (SAT). For our method, we list the runtime for computing FTLE, the distance measures, and the final linear optimization. All measurements are in seconds.

	Baseline [13]		Our method		
Dataset	SAT on	SAT off	FTLE	distances	optimize
TURB. 1	85.47	118.34	2402.77	1407.15	106.6
TURB. 2	77.95	108.63	2402.77	1820.09	116.29
TURB. 3	62.75	109.68	2402.77	1720.46	110.83
TURB. 4	65.52	104.54	2402.77	1723.24	108.01
TURB. 5	64.64	91.77	1928.22	1381.09	100.09

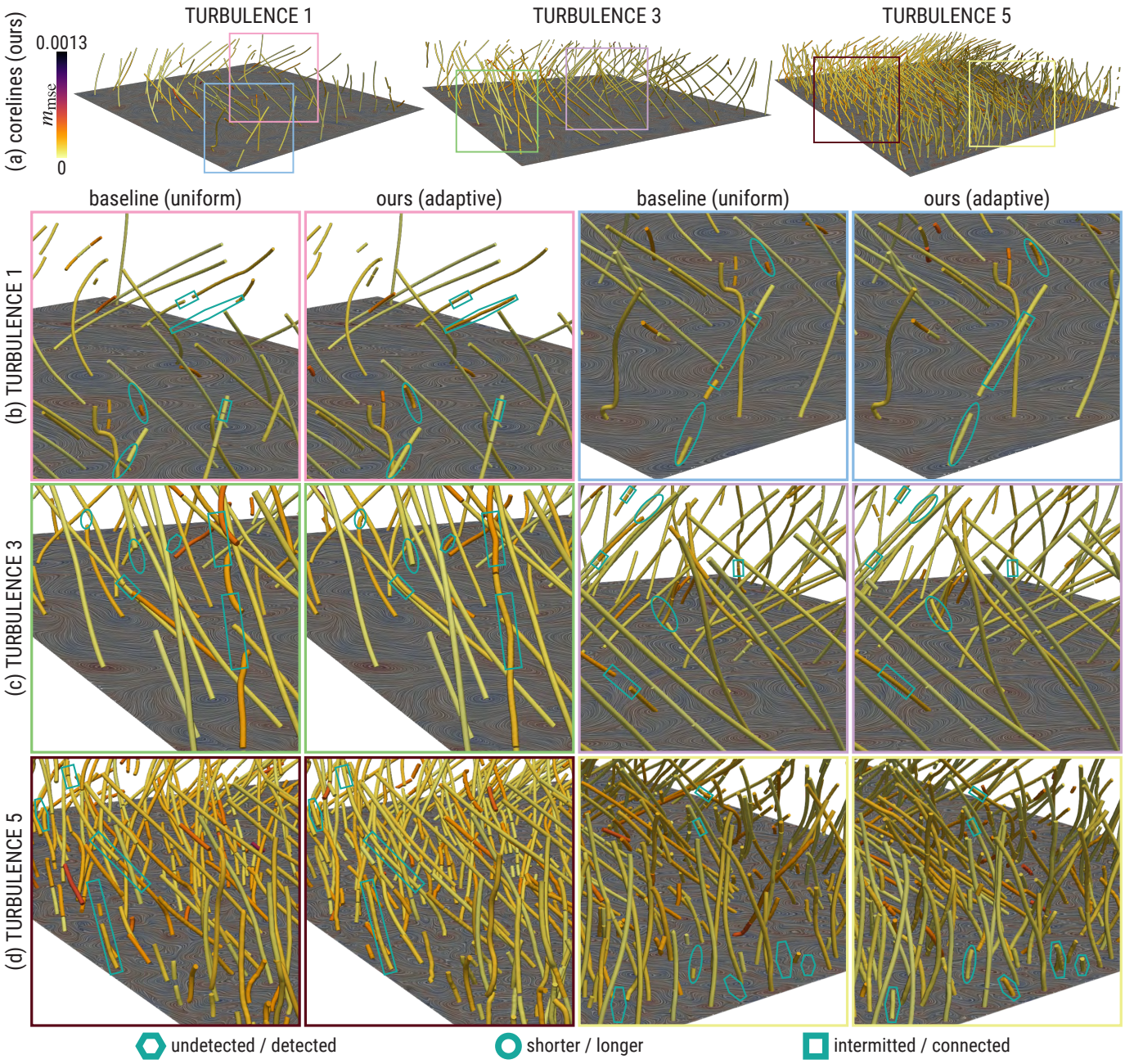

Fig. 8: **Coreline comparisons** for increasingly turbulent flow fields. (a) corelines (ours) from TURBULENCE1 (left), TURBULENCE3 (middle), and TURBULENCE5 (right), where the coreline color encodes the mean segment error per coreline. (b) TURBULENCE1 corelines zoom in, where the baseline [13] results are on the left and our results are on the right for each pair of insets. (c) TURBULENCE3 zoom ins. (d) TURBULENCE5 zoom ins.

is only possible with uniform weights, and cannot be applied in our method. We implemented the extraction of FTLE and the guided distances on the GPU using CUDA. The FTLE field is extracted at three times the spatial resolution of the underlying vector field, which takes about 6 seconds per time slice (forward and backward).

5 CONCLUSIONS

Reference frame optimization methods are a promising approach to vortex coreline extraction in time-dependent flow, as they enable the application of techniques originally developed for steady vector fields [13]. However, all existing local reference frame optimization approaches [2, 13, 18, 19, 44] find the optimal reference frame for a spatial neighborhood in which the flow is assumed to move coherently. In turbulent flow, however, this assumption does not hold, since multiple vortices can exist in close proximity. Nevertheless, the vortices are separated by transport barriers that can be approximated as ridges in the finite-time Lyapunov exponent (FTLE) field. Using the FTLE field, we locally adjust the neighborhood in which the reference frame is optimized by adapting the corresponding weighting, which enables the extraction of nearby vortices.

Currently, our approach is targeted to 2D flows. Extending the use of guidance fields to constrain 3D local reference frame optimization to coherent flow regions is a promising research direction. Besides, previous reference frame optimizations have been temporally local, i.e., they only needed the velocity field and its time derivative in a single timestep. In our approach, however, since we use the FTLE field to guide the adaptive neighborhood weighting, we now require larger time windows. Furthermore, the FTLE computation increases the overall runtime of the algorithm. In the future, it will be interesting to explore temporally local alternatives for the determination of transport barriers.

Acknowledgments

We thank Paul Himmler for simulating the TURBULENCE data sets. This work was partially supported by DFG grant no. GU 945/7-1. This work was supported by King Abdullah University of Science and Technology (KAUST) baseline funding.

References

[1] I. Baeza Rojo, M. Gross, and T. Günther. Accelerated Monte Carlo rendering of finite-time Lyapunov exponents. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):708–718, 2020. doi: 10.1109/TVCG.2019.2934313 2

[2] I. Baeza Rojo and T. Günther. Vector field topology of time-dependent flows in a steady reference frame. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):280–290, 2020. doi: 10.1109/TVCG.2019.2934375 2, 3, 9

[3] S. Barakat, C. Garth, and X. Tricoche. Interactive computation and rendering of finite-time Lyapunov exponent fields. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1368–1380, 2012. doi: 10.1109/TVCG.2012.33 2

[4] S. Barakat and X. Tricoche. Adaptive refinement of the flow map using sparse samples. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2753–2762, 2013. doi: 10.1109/TVCG.2013.128 2

[5] R. Bujack. Discussion and visualization of distinguished hyperbolic trajectories as a generalization of critical points to 2D time-dependent flow. In *Topological Data Analysis and Visualization (TopoInVis)*, pp. 59–69. IEEE, Oklahoma City, OK, USA, 2022. doi: 10.1109/TopoInVis57755.2022.00013 2

[6] R. Bujack, L. Yan, I. Hotz, C. Garth, and B. Wang. State of the art in time-dependent flow topology: Interpreting physical meaningfulness through mathematical properties. *Computer Graphics Forum*, 39(3):811–835, 2020. doi: 10.1111/cgf.14037 1, 2

[7] G. E. Fasshauer. *Meshfree Approximation Methods with MATLAB*. World Scientific Publishing Company, 2007. doi: 10.1142/6437 3

[8] S. Fleishman, D. Cohen-Or, and C. T. Silva. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics*, 24(3):544–552, 2005. doi: 10.1145/1073204.1073227 3

[9] A. Friederici, H. T. Mahamadou Kele, I. Hoteit, T. Weinkauf, H. Theisel, and M. Hadwiger. A Lagrangian method for extracting eddy boundaries in the Red Sea and the Gulf of Aden. In *IEEE Scientific Visualization Conference (SciVis)*, pp. 52–56. IEEE, Berlin, Germany, 2018. doi: 10.1109/SciVis.2018.8823600 2

[10] A. Friederici, H. Theisel, and T. Günther. Trajectory vorticity - computation and visualization of rotational trajectory behavior in an objective way. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–14, 2024. doi: 10.1109/TVCG.2024.3421555 3

[11] G. Froyland and K. Padberg-Gehle. Almost-invariant and finite-time coherent sets: Directionality, duration, and diffusion. In *Ergodic Theory, Open Dynamics, and Coherent Structures*, pp. 171–216. Springer New York, New York, NY, 2014. doi: 10.1007/978-1-4939-0419-8_9 2

[12] T. Günther and I. Baeza Rojo. Introduction to vector field topology. In *Topological Methods in Data Analysis and Visualization VI*, pp. 289–326. Springer, Cham, Nyköping, Sweden, 2021. doi: 10.1007/978-3-030-83500-2_15 2

[13] T. Günther, M. Gross, and H. Theisel. Generic objective vortices for flow visualization. *ACM Transactions on Graphics*, 36(4):141:1–141:11, 2017. doi: 10.1145/3072959.3073684 1, 3, 6, 7, 8, 9

[14] T. Günther, A. Kuhn, and H. Theisel. MCFTLE: Monte Carlo rendering of finite-time Lyapunov exponent fields. *Computer Graphics Forum (Proc. EuroVis)*, 35(3):381–390, 2016. doi: 10.1111/cgf.12914 2

[15] T. Günther, M. Schulze, and H. Theisel. Rotation invariant vortices for flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):817–826, 2016. doi: 10.1109/TVCG.2015.2467200 3

[16] T. Günther and H. Theisel. The state of the art in vortex extraction. *Computer Graphics Forum*, 37(6):149–173, 2018. doi: 10.1111/cgf.13319 2, 6, 8

[17] T. Günther and H. Theisel. Objective vortex corelines of finite-sized objects in fluid flows. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):956–966, 2019. doi: 10.1109/TVCG.2018.2864828 3

[18] T. Günther and H. Theisel. Hyper-objective vortices. *IEEE Transactions on Visualization and Computer Graphics*, 26(3):1532–1547, 2020. doi: 10.1109/TVCG.2018.2868760 3, 9

[19] T. Günther and H. Theisel. Objective Lagrangian vortex cores and their visual representations. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):76–85, 2025. doi: 10.1109/TVCG.2024.3456384 3, 9

[20] M. Hadwiger, M. Mlejnek, T. Theußl, and P. Rautek. Time-dependent flow seen through approximate observer killing fields. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1257–1266, 2019. doi: 10.1109/TVCG.2018.2864839 1, 3, 4

[21] M. Hadwiger, T. Theußl, and P. Rautek. Riemannian geometry for scientific visualization. *SIGGRAPH Asia 2022 courses*, (5), 2022. doi: 10.1145/3550495.3558227 4

[22] G. Haller. Lagrangian coherent structures from approximate velocity data. *Physics of Fluids*, 14(6):1851–1861, 2002. doi: 10.1063/1.1477449 2, 6

[23] G. Haller. An objective definition of a vortex. *Journal of Fluid Mechanics*, 525:1–26, 2005. doi: 10.1017/S0022112004002526 2, 3

[24] G. Haller. Lagrangian coherent structures. *Annual Review of Fluid Mechanics*, 47:137–162, 2015. doi: 10.1146/annurev-fluid-010313-141322 2

[25] J. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *Computer*, 22(8):27–36, 1989. doi: 10.1109/2.35197 2

[26] J. L. Helman and L. Hesselink. Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications*, 11(3):36–46, 1991. doi: 10.1109/38.79452 2

[27] L. Hofmann and F. Sadlo. Extraction of distinguished hyperbolic trajectories for 2D time-dependent vector field topology. *Computer Graphics Forum*, 39(3):303–316, 2020. doi: 10.1111/cgf.13982 2

[28] L. Hofmann and F. Sadlo. Local extraction of 3D time-dependent vector field topology. *Computer Graphics Forum*, 40(3):111–122, 2021. doi: 10.1111/cgf.14293 2

[29] J. Kasten, C. Petz, I. Hotz, B. Noack, and H.-C. Hege. Localized finite-time Lyapunov exponent for unsteady flow analysis. In *Proceedings of Vision, Modeling and Visualization Workshop*, pp. 265–276, 2009. 2

[30] J. Kasten, J. Reininghaus, I. Hotz, and H.-C. Hege. Two-dimensional time-dependent vortex regions based on the acceleration magnitude. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE SciVis)*, 17(12):2080–2087, 2011. doi: 10.1109/TVCG.2011.249 2

[31] B. Kaszás, T. Pedergnana, and G. Haller. The objective deformation component of a velocity field. *European Journal of Mechanics - B/Fluids*, 98:211–223, 2023. doi: 10.1016/j.euromechflu.2022.12.007 3

[32] B. Kim and T. Günther. Robust reference frame extraction from unsteady 2D vector fields with convolutional neural networks. *Computer Graphics Forum*, 38(3):285–295, 2019. doi: 10.1111/cgf.13689 3

[33] A. Kuhn, C. Rössl, T. Weinkauf, and H. Theisel. A benchmark for evaluating FTLE computations. In *Proceedings of the 5th IEEE Pacific Visualization Symposium*, pp. 121–128. IEEE, Songdo, Korea, 2012. doi: 10.1109/PacificVis.2012.6183582 2

[34] P. Lancaster and K. Salkauskas. Surfaces generated by moving least squares methods. *Mathematics of Computation*, 37(155):141–158, 1981. doi: 10.1090/S0025-5718-1981-0616367-1 3

[35] D. Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67(224):1517–1531, 1998. doi: 10.1090/S0025-5718-98-00974-0 3

[36] Y. Lipman. Stable moving least-squares. *Journal of Approximation Theory*, 161(1):371–384, 2009. doi: 10.1016/j.jat.2008.10.011 3

[37] H. J. Lugt. The dilemma of defining a vortex. In *Recent Developments in Theoretical and Experimental Fluid Mechanics*, pp. 309–321. Springer, 1979. doi: 10.1007/978-3-642-67220-0_32 3

[38] S. Meckel, M. Markl, and S. Wetzel. Identification of vortex cores in cerebral aneurysms on 4D flow MRI. *American Journal of Neuroradiology*, 41(5):E26, 2020. doi: 10.3174/ajnr.A6480 2

[39] A. Nealen. An as-short-as-possible introduction to the least squares, weighted least squares and moving least squares methods for scattered data approximation and interpolation, 2004. 3, 4

[40] B. Nsonga, M. Niemann, J. Fröhlich, J. Staib, S. Gumhold, and G. Scheuermann. Detection and visualization of splat and antisplat events in turbulent flows. *IEEE Transactions on Visualization and Computer Graphics*, 26(11):3147–3162, 2020. doi: 10.1109/TVCG.2019.2920157 2

[41] S. Popinet. The basilisk code. http://basilisk.fr/ (accessed: 2025-03-31). 6

[42] S. Popinet. Free computational fluid dynamics. *ClusterWorld*, 2(6), 2004. 6

[43] P. Rautek, M. Mlejnek, J. Beyer, J. Troidl, H. Pfister, T. Theußl, and M. Hadwiger. Objective observer-relative flow visualization in curved spaces for unsteady 2D geophysical flows. *IEEE Transactions on Visualiza-*

tion and Computer Graphics, 27(2):283–293, 2021. doi: 10.1109/TVCG.2020.3030454 3

[44] P. Rautek, X. Zhang, B. Woschizka, T. Theußl, and M. Hadwiger. Vortex lens: Interactive vortex core line extraction using observed line integral convolution. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):55–65, 2024. doi: 10.1109/TVCG.2023.3326915 3, 9

[45] M. Roth and R. Peikert. Flow visualization for turbomachinery design. In *Proceedings of Seventh Annual IEEE Visualization*, pp. 381–384. IEEE, San Francisco, CA, USA, 1996. doi: 10.1109/VISUAL.1996.568137 2

[46] F. Sadlo and R. Peikert. Efficient visualization of Lagrangian coherent structures by filtered AMR ridge extraction. *IEEE Trans. on Visualization and Computer Graphics (Proc. IEEE VIS)*, 13(6):1456–1463, 2007. doi: 10.1109/TVCG.2007.70554 2

[47] F. Sadlo and D. Weiskopf. Time-dependent 2-D vector field topology: An approach inspired by Lagrangian coherent structures. *Computer Graphics Forum*, 29(1):88–100, 2010. doi: 10.1111/j.1467-8659.2009.01546.x 2

[48] S. C. Shadden, F. Lekien, and J. E. Marsden. Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena*, 212(3):271–304, 2005. doi: 10.1016/j.physd.2005.10.007 2

[49] D. Stelter, T. Wilde, C. Rössl, and H. Theisel. A particle-based approach to extract dynamic 3D FTLE ridge geometry. *Computer Graphics Forum*, 44(1):e15203, 2025. doi: 10.1111/cgf.15203 2

[50] H. Theisel, M. Hadwiger, P. Rautek, T. Theußl, and T. Günther. Vortex criteria can be objectivized by unsteadiness minimization. *Physics of Fluids*, 33(10), 2021. doi: 10.1063/5.0063817 3

[51] H. Theisel and H.-P. Seidel. Feature flow fields. In *Proceedings of the Eurographics / IEEE VGTC Symposium on Visualization*, pp. 141–148. The Eurographics Association, Grenoble, France, 2003. doi: 10.2312/VisSym/VisSym03/141-148 2

[52] A. Tikhonov and V. Arsenin. *Solutions of Ill-posed Problems*. Winston, 1977. 5

[53] C. Truesdell and W. Noll. *The Non-Linear Field Theories of Mechanics*. Springer, Berlin, Heidelberg, 1965. doi: 10.1007/978-3-642-46015-9_1 1, 2

[54] M. Üffinger, F. Sadlo, and T. Ertl. A time-dependent vector field topology based on streak surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):379–392, 2013. doi: 10.1109/TVCG.2012.131 2

[55] T. Weinkauf. *Extraction of Topological Structures in 2D and 3D Vector Fields*. PhD thesis, University Magdeburg, 2008. https://opendata.uni-halle.de/bitstream/1981185920/10919/1/tinweinkauf.pdf. 2

[56] H. Wendland. *Scattered Data Approximation*. Cambridge University Press, 2004. doi: 10.1017/CBO9780511617539 3, 4

[57] T. Wilde, C. Rössl, and H. Theisel. FTLE ridge lines for long integration times. In *IEEE Scientific Visualization Conference (SciVis)*, pp. 57–61. IEEE, Berlin, Germany, 2018. doi: 10.1109/SciVis.2018.8823761 2

[58] T. Wilde, C. Rössl, and H. Theisel. Recirculation surfaces for flow visualization. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Scientific Visualization 2018)*, 25(1):946–955, 2019. doi: 10.1109/TVCG.2018.2864813 2

[59] L. Yan, P. A. Ullrich, L. P. Van Roekel, B. Wang, and H. Guo. Multilevel robustness for 2D vector field feature tracking, selection and comparison. *Computer Graphics Forum*, 42(6):e14799, 2023. doi: 10.1111/cgf.14799 2

[60] X. Zhang, M. Hadwiger, T. Theußl, and P. Rautek. Interactive exploration of physically-observable objective vortices in unsteady 2D flow. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE VIS 2021)*, 28(1):281–290, 2022. doi: 10.1109/TVCG.2021.3115565 3, 4

[61] X. Zhang, P. Rautek, and M. Hadwiger. Vortextransformer: End-to-end objective vortex detection in 2D unsteady flow using transformers. *Computer Graphics Forum (Proc. Eurographics)*, 44(2):e70042, 2025. doi: 10.1111/cgf.70042 3

[62] X. Zhang, P. Rautek, T. Theußl, and M. Hadwiger. Enhancing Material Boundary Visualizations in 2D Unsteady Flow through Local Reference Frame Transformations. *Computer Graphics Forum (Proc. Eurovis 2025)*, 2025. doi: 10.1111/cgf.70128 3