Vision, Modeling, and Visualization (2023)
T. Grosch and M. Guthe (Eds.)

Autonomous Particles for In-Situ-Friendly Flow Map Sampling

S. Wolligandt, C. Rössl, C. Chi, D. Thévenin, H. Theisel

University of Magdeburg, Germany

Abstract

Computing and storing flow maps is a common approach to processing and analyzing large flow simulations in a Lagrangian way. Accurate Lagrangian-based visualizations require a good sampling of the flow map. We present an In-Situ-friendly flow map sampling strategy for flows using Autonomous Particles that do not need information of neighboring particles: they can be advected individually without knowing about each other. The main idea is to observe a linear neighborhood of a particle during advection. As soon as the neighborhood cannot be considered linear anymore, an adaptive splitting is performed. For observing the linear neighborhood, each particle is equipped with an ellipsoid that also gets advected by the flow. By splitting these ellipsoids into smaller ones in regions of non-linear behavior, critical and more interesting regions of the flow map are more densely sampled. Our sampling approach uses only forward integration and no adaptive integration from the past. This makes it applicable in and well-suited for in In-Situ environments. We compare our approach to existing sampling techniques and apply it to several artificial and real data sets.

1. Introduction

The standard representation of flows are time-dependent velocity fields stemming from measurements or numerical simulations. From these flow fields, particle trajectories are typically obtained by numerical integration. In recent years, an alternative representation of flows has attracted attention: flow maps. Flow maps explicitly encode the location of particles after integrating them for a certain finite time. This qualifies flow maps for the analysis of Lagrangian structures in flows. However, flow maps are hard to handle because they are high-dimensional objects, expensive in computation and storage, and tend to have large gradients. This makes flow map sampling a challenging task, in particular, if the reconstruction from samples is to be used as a tool in applications.

If a flow simulation is performed from a Lagrangian perspective (e.g. by SPH), the standard output consists of a set of particle trajectories, each of which is a sample of the flow map. If an Eulerian simulation is performed (e.g., by numerically solving the Navier-Stokes equations on a grid), its output is a discrete velocity field. For this case, the problem is to find a good flow map sampling from the velocity field.

Nowadays Eulerian flow simulation faces the problem that storing the output (i.e., the discrete velocity field represented at the simulation resolution) is more expensive – both, in time and storage space – than the actual simulation. For this reason, typically only a small subset of the velocity field (e.g. every 1 000th time step) is written to disk – the remaining steps are discarded. This raises the problem of accuracy of particle integration in the stored velocity field. A popular way to deal with this is an *in-situ* approach: local operations on the velocity field (e.g., particle integration) are carried out *during simulation*: immediately after the part of the velocity field has been simulated and before it is discarded. This enables an exact computation of particle trajectories (up to the accuracy of the numerical integration) on the high-resolution simulated velocity field. This accuracy boost is the main motivation for flow map sampling, particularly in in-situ environments.

Our approach can also be beneficial if the data analysis is not performed in-situ but as a post-process: Even if that all simulation results, i.e., velocity fields for all time steps, can be stored, the amount of data may be huge and expensive to load from storage. In this case, the only option for analysis is a streaming approach with a sliding window of time steps in main memory. From an algorithmic perspective, this situation is similar to the in-situ scenario and fits perfectly into our setting.

Given a time-dependent velocity field $\mathbf{v}(\mathbf{x},t)$ in the spatial domain D and the time domain $T=[t_s,t_e]$ with $t_s<t_e$, a *flow map sampling* S is a finite set of pairs of locations

$$S=\{(\mathbf{x}_i,\mathbf{y}_i):\mathbf{x}_i\in D,\mathbf{y}_i=\phi_{t_s}^{t_e-t_s}(\mathbf{x}_i),\ \ i=1,\dots,n\}\ ,$$

where $\phi_t^\tau(\mathbf{x})$ denotes the flow map, i.e., the location of a particle starting at $\mathbf{x}$ at time t after integrating $\mathbf{v}$ over a finite period τ.

Flow map processing consists of two parts:

Flow map sampling: we search for a "good" sampling S of the flow map for a certain budget (e.g., number of particles or computation time) relative to the total "cost" of the simulation. Typically only a small fraction of the computing time can be allocated for flow map sampling.

Flow map reconstruction: From a sampling S, we approximate the flow map $\phi_{t_s}^{t_e-t_s}(\mathbf{x})$ for an arbitrary source location $\mathbf{x}$. We call this *forward* flow map reconstruction. In contrast, *backward* flow map

DOI: 10.2312/vmv.20231242

reconstruction approximates the flow map $\phi_{t_e}^{t_s-t_e}(\mathbf{y})$ for an destination location $\mathbf{y}$. Note that both types of reconstruction are relevant for flow analysis, because the backward direction answers the question where a particle at a certain location came from.

Flow map reconstruction refers to estimating the flow map from the samples for an arbitrary domain point, typically by some interpolation. Most applications of flow map sampling require a reconstruction step. For example, to evaluate the quality of our flow map sampling, we measure the reconstruction error on a regular grid. Note also that flow map sampling should be computed in an in-situ environment because of the increased integration accuracy. Contrarily, flow map reconstruction can be done as post-processing on the stored sampling S.

In this paper, we present a new approach to flow map sampling that is evaluated on a state-of-the-art flow map reconstruction technique [BT13] for both forward and backward reconstruction.We postulate that the following properties are desired for the construction of a flow map sampling S:

(1.) *Adaptivity*: Since the flow map tends to have strong gradients, adaptive samplings S are likely to produce less reconstruction error. In areas of strong flow map gradients, a higher sampling density is desired.

(2.) *Only forward integration*: Even though we also want to reconstruct backward flow maps (i.e., flow maps over the negative time interval $t_s - t_e$), for the creation of the sampling S we allow only forward integration. If the simulation of $\mathbf{v}$ is large and complex, the computation must be performed in-situ by streaming the data: At any moment, we only have a small "window" of time steps of the data available, and the simulation window is always moving forward in time. Therefore, allowing backward integration means that the sampling is not in-situ-friendly, i.e., it cannot be carried out in an in-situ environment. For the reconstruction, however, a backward flow map is considered because backward Lagrangian analysis gives additional information about the flow and is frequently done in flow analysis.

(3.) *No integration from the past*: Once during the construction of S a particle is forward integrated until a certain time t, it is not allowed to start a new particle integration (e.g., for gradient estimation) from a time $t' < t$. This property is also necessary to get in-situ-friendliness since in an in-situ environment each time slab is only available once and then discarded.

(4.) *Autonomous particles*: During the integration of particles, decisions about events like splitting, merging, birth, death or particles are necessary to achieve adaptivity. The usual approach considers the neighborhood of particles and tracks their spatial adjacency. Efficient adjacency queries and their temporal updates require additional data structures. Although, there exist efficient solutions (e.g., k-d-trees or spatial hashing), these data structures must be available "globally". This makes a difference for parallel processing (e.g., with domain decomposition): any processor must be able to access and mutate information about particle connectivity, which leads to communication overhead (exclusively for the reconstruction). As an alternative, we demand that all particles are *autonomous*, i.e., they do not have or need any information about their neighborhood.

(5.) *3D*: The flow map sampling techniques should be applicable for 2D and 3D time-dependent velocity fields.

We consider all these five properties essential for a useful algorithm for flow map sampling. Without *adaptivity*, we do not expect sufficient sampling accuracy in flows with a strong flow map gradient. *Forward integration* and *no dependence on the past* are necessary for in-situ friendliness, i.e., without them, the sampling cannot be done in an in-situ environment and is therefore restricted to rather small data sets. *Autonomous particles* ensure that a strong parallelization of the sampling algorithm is possible. Moreover, the mode of parallel computation is defined entirely by the simulation, there will be no additional constraints from the reconstruction. In an extreme scenario, this allows assigning each particle its own processor because no communication between particles is necessary at all.

2. Related Work

In this section, we review prior work that is related to our method. While our main contribution is an approach to flow map sampling, a reconstruction scheme is required for evaluation.

Flow map sampling. Generally, efficient sampling requires guidance such that samples are concentrated in regions with high detail and fewer samples are spent on homogeneous regions: the sampling rate should be adapted to the local detail. In the following, we review flow map sampling w.r.t. to the postulated desired properties (1.)-(5.). Table 1 shows a summary.

Garth et al. [GGTH07] present a method for adaptive computation of FTLE from 2D and 3D flow fields. Sadlo and Peikert [SP07] aim at ridge extraction to identify Lagrangian Coherent Structures (LCS) using an adaptive mesh representation, and Hlawatsch et al. [HSW11] sample trajectories adaptively using a hierarchical approach. All these three approaches require information about neighborhoods and may require to start new integration from the past (i.e, they ✗ lack (3.) and (4.)).

Agranovsky et al. [ACG*14] aim at an in-situ sampling of the flow map that can be reconstructed and analyzed in a post-process. We share the same objective, however, taking a different approach: They start from regular seeds, i.e., a regular sampling, that are advected in the flow. Whenever the (exponential) separation has grown too much after a period, they re-initiate the regularly placed seeds and iterate this process. This process can be interpreted as adapting in the temporal domain by reseeding, but there is no adaptation in the spatial domain (i.e., it ✗ lacks (1.)).

Kuhn et al. [KER*14] advect a 2D triangulation (i.e., ✗ lacks (4.) and (5.)) and record certain events like the change of orientation of a triangle to extract LCS from sparse samples of trajectories. Rapp et al. [RPD20] take a statistical approach to find an in some sense optimal sampling of trajectories. This requires the use of a global clustering and thus a notion of neighborhood and it may place new samples (i.e., it ✗ lacks (3.) and (4.)).

Flow map reconstruction. Flow map reconstruction methods estimate continuous flow maps from discrete samples for purposes

	(1.)	(2.)	(3.)	(4.)	(5.)
regular sampling	✗	✓	✓	✓	✓
[GGTH07]	✓	✓	✗	✗	✓
[SP07]	✓	✓	✗	✗	✓
[HSW11]	✓	✓	✗	✗	✓
[ACG*14]	✗	✓	✓	✓	✓
[KER*14]	✓	✓	✓	✗	✗
[RPD20]	✓	✓	✗	✗	✓
our approach	✓	✓	✓	✓	✓

Table 1: *Summary of desired properties fulfilled by regular samples, related work and our method.*

such as resampling, evaluation from efficient or external storage, or substituting expensive numerical integration.

Chandler et al. [COJ15] replace numerical integration by interpolation in certain regions, an error analysis is provided in [CBJ16]. Agranovsky et al. [AOGJ15] provide a multi-resolution representation for a set of trajectories that enables efficient loading the data from external storage into main memory. Bujack and Joy [BJ15] evaluate representations of polygonal trajectories as parametric curves, and Hummel et al. [HBJG16] analyze the Lagrangian error of trajectories.

Our method provides an adaptive sampling of the flow map without tracking information of particle neighborhoods. For flow map reconstruction, we need to be able to compute – i.e., interpolate – the flow map at any domain point, e.g., for resampling the resulting flow map on a dense regular grid. This is essentially a *scattered data interpolation* problem.

Barakat and Tricoche [BT13] present a variant of Sibson's natural neighbor interpolation [Sib81] that is tailored to flow map reconstruction. Our flow map reconstruction is based on their approach. Natural neighbor interpolation requires a partition of the domain into Voronoi cells. We use the CGAL library [The22] for implementation.

Machine-learning approaches for reconstruction. In recent years, machine learning methods, particularly those based on deep-learning, have been used for flow reconstruction. There is a similar trend for supporting flow simulation.

Guo et al. [GYH*20] aim at spatial super-resolution for vector fields. Sahoo et al. [SB21] additionally consider loss of accuracy for advection in the flow. Jabob et al. [JGG21] present a "neural flow map interpolation" that is based on learning from 8000 2D unsteady flows with a total amount of 16TB of training data. Gu et al. [GHCW21] reconstruct unsteady flows from incomplete information: representative streamlines.

All these approaches give promising results. However, they require learning from a possibly huge set of training data. Our method could certainly benefit from such learning-based interpolation in the reconstruction phase. We prefer the above-mentioned natural neighbor interpolation [BT13] as a standard approach for three reasons: firstly, it is much simpler (and possibly more efficient), secondly, the quality of the reconstruction is good enough. And last not least, interpolation based on machine learning may provide good results even for bad input. Here, the input comes from our flow map sampling, and we must make sure that the interpolation does not "distort" our results in a sense of an "unfair a posteriori improvement".

Autonomous particles and ghost particles. Engelke et al. [ELPH19] use the term *autonomous particles* in a similar but different context: They present an interactive tool for the visual analysis of flow data. Their method applies a particle system and advection for adaptive sampling and feature detection. The term *autonomous* is used for particles that are advected independently and possibly in parallel without knowledge about "neighboring" particles and events like death and split events are guided by importance fields. The objective of the method is a visual analysis by concentrating samples in regions of interest. While this can certainly be classified as adaptive sampling, it does neither aim at nor provide a reconstruction of the flow map. In particular, the sampling is based on external information like importance or guidance fields and explicitly ignores gradient information.

Oster at al. [OAM*18] use *ghost particles* for tracking and reconstruction of flame fronts in a combustion process. The custom simulation is expensive and conducted on a super-computer, the tracking of the flame fronts is interleaved as an in-situ analysis with minimal impact on the simulation in terms of, e.g., computational cost or change of implementation. The *ghost particles* are used to estimate flow map gradients, and the gradient steers split and merge events. The objective is to provide a sampling of the tangent space of the flame front. Our method similarly makes use of ghost particles for gradient estimation and uses this information for defining split events, however, in a different way and with a different objective in mind.

3. Our method

Our method is based on the integration of an autonomous particle with the simultaneous observation of its neighborhood: During the integration of a particle, we also observe its neighborhood. We ensure that the neighborhood is small enough to be considered linear, i.e., the supposed advection of any particle within this region can be faithfully approximated by a linear map. If during the integration the neighborhood cannot be considered linear anymore, it is split into multiple smaller regions. A neighborhood around a particle can be considered (approximately) linear if during advection a unit sphere deforms to an (approximate) ellipsoid with the particle in its center. In this case, the neighborhood can be considered a "safe space": accurate reconstruction inside the neighborhood is possible solely from the flow map and its gradient. Both are computed independently for a single – autonomous particle – none of the other particles computed for sampling the flow map necessary.

We introduce our approach generally for 3D time-dependent flows. 2D flows appear as a special case, there is no conceptual difference. For sake of simplicity, we prefer using the terms *ellipse* and *circle* instead of *ellipsoid* and *sphere* and explain with figures showing the 2D case.

3.1. Particles and deforming neighborhoods

Given is a particle $\mathbf{x}_0$ at a time t_0. Its new position after advection for a period τ is given by the *flow map* $\phi(\mathbf{x}_0, t_0, \tau)$. We equip the

particle with an ellipsoidal neighborhood around $\mathbf{x}_0$ represented by a symmetric positive definite matrix $\mathbf{S}_0$: Then the neighborhood is bounded either by the image of the parametric surface $\mathbf{x}_0 + \mathbf{S}_0\,\mathbf{r}$ with $\|\mathbf{r}\| = 1$, or implicitly as the point set that satisfies $(\mathbf{x}-\mathbf{x}_0)^T \mathbf{S}_0^{-2} (\mathbf{x}-\mathbf{x}_0) = 1$. In the following, we treat the matrix representation $\mathbf{S}_0$ as a synonym for the ellipsoidal neighborhood.

The advection of the particle $\mathbf{x}_0$ and its neighborhood $\mathbf{S}_0$ leads to a deformation of $\mathbf{S}_0$. If we can assume a linear flow map within $\mathbf{S}_0$, we can use the spatial gradient $\nabla\phi := \nabla\phi(\mathbf{x}_0, t_0, \tau)$ to approximate the advection by a period τ as

$$\phi(\mathbf{x}, t_0, \tau) \;=\; \phi(\mathbf{x}_0, t_0, \tau) \;+\; \nabla\phi \cdot (\mathbf{x}-\mathbf{x}_0)\;.$$

Hence, the advection of the ellipsoidal region $\mathbf{S}_0$ yields an ellipsoidal neighborhood $\mathbf{S}(\tau)$ such that

$$\mathbf{S}(\tau)^2 = \nabla\phi\;\mathbf{S}_0{}^2\;(\nabla\phi)^T. \tag{1}$$

Note that this holds only under the assumption of a linear flow map inside $\mathbf{S}_0$. If this is not the case, advection deforms the ellipsoidal region $\mathbf{S}_0$ to any arbitrary shape.

Given is a particle $\mathbf{x}_0$ at a time t_0. The standard method for numerical estimation of the flow map gradient is to integrate additional auxiliary "ghost" particles. They are usually seeded from $\mathbf{x}_0$ at positions shifted into the direction of the coordinate axes. We use a similar approach in a local coordinate frame: Given the additional neighborhood matrix $\mathbf{S}_0$ around $\mathbf{x}_0$, we seed ghost particles shifted in the direction of the eigenvectors of $\mathbf{S}_0$ to estimate the flow map gradient. Note that the use of ghost particles does not violate any of the postulated properties: integration is still autonomous and independent of other particles, and we integrate strictly forward in time.

Consider the spectral decomposition of $\mathbf{S}_0$ with eigenvectors as columns of the orthogonal matrix $\mathbf{Q}_0$ and the diagonal matrix Σ_0 with the positive eigenvalues $\mu_1 \geq \mu_2 \geq \mu_3 > 0$ such that

$$\mathbf{S}_0 = \mathbf{Q}_0\,\Sigma_0\,\mathbf{Q}_0{}^T. \tag{2}$$

We define ghost points $\mathbf{x}_\mathbf{k}$ in the neighborhood of a point $\mathbf{x}_0$ as

$$\mathbf{x}_\mathbf{k} = \mathbf{x}_0 + \mathbf{Q}_0\,\Sigma_0\,\mathbf{k}$$

for a vector $||\mathbf{k}|| = 1$. From the ghost points, we integrate particles and obtain the flow maps

$$\phi = \phi(\mathbf{x}_0, t_0, \tau) \qquad \text{and} \qquad \phi_\mathbf{k} = \phi(\mathbf{x}_\mathbf{k}, t_0, \tau)\;.$$

For $i = 1,2,3$, we define central differences

$$\mathbf{h}_i = \frac{1}{2}\,(\phi_{\mathbf{e}_i} - \phi_{-\mathbf{e}_i})\;,$$

where $\mathbf{e}_1 = (1,0,0), \mathbf{e}_2 = (0,1,0), \mathbf{e}_3 = (0,0,1)$ denote the canonical basis, and $\mathbf{H}$ is the the matrix with $\mathbf{h}_i$ as columns:

$$\mathbf{H}(\tau) = (\mathbf{h}_1,\, \mathbf{h}_2,\, \mathbf{h}_3)\,. \tag{3}$$

Applying central differences in the local coordinate system of eigenvectors provides an approximation of the flow map gradient as

$$\nabla\phi(\mathbf{x}_0, t_0, \tau) \;=\; \mathbf{H}(\tau)\;{\Sigma_0}^{-1}\;\mathbf{Q}_0{}^T. \tag{4}$$

By substituting (2) and (4) into (1), we obtain an approximation of

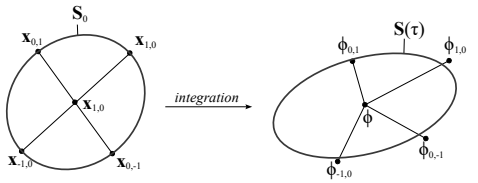

Figure 1: *Particle and inner neighborhood during integration.*

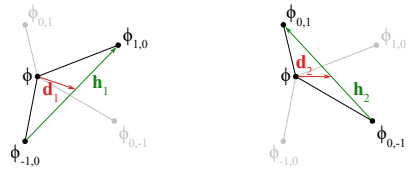

Figure 2: *Linearity measure (6) illustrated in 2D:* $\mathbf{d}_i = \frac{1}{2}\,(\phi_{\mathbf{e}_i} + \phi_{-\mathbf{e}_i}) - \phi$ *denotes the deviation from mean term (Laplace), which vanishes in a perfectly linear neighborhood. The magnitude of the "spread"* $\mathbf{h}_i$ *provides the baseline for a relative error.*

$\mathbf{S}(\tau)$ by computing its square

$$\mathbf{S}(\tau)^2 = \mathbf{H}(\tau)\;\mathbf{H}(\tau)^T. \tag{5}$$

Figure 1 illustrates the setup. The main idea of our approach is to observe the distortion of $\mathbf{S}(\tau)$ during the advection. As soon as advection cannot be approximated by a linear map anymore (i.e., the advection of the region $\mathbf{S}(\tau)$ is no more an ellipsoid), we subdivide the particle.

3.2. Measuring the linearity of the flow map

We define the error term

$$\mathrm{err}_{\mathrm{nl}}(\tau) \;=\; \sum_i \frac{||\frac{1}{2}\,(\phi_{\mathbf{e}_i} + \phi_{-\mathbf{e}_i}) - \phi||^2}{||\mathbf{h}||^2}\;. \tag{6}$$

The smaller $\mathrm{err}_{\mathrm{nl}}$, the more we can assume a linear behavior of the flow map in $\mathbf{S}$. The term measures a relative error: The nominators measure "linearity" by computing a deviation from mean (i.e., a discrete univariate Laplace operator should yield zero), and the denominators relate the absolute values to the local magnitude of "spread" in the flow map. Note also that $\mathrm{err}_{\mathrm{nl}}$ is independent of the size of the neighborhood, because uniform scaling of ϕ or $\phi_\mathbf{k}$ does not change the error. Figure 2 illustrates this setting.

In addition to linearity, we measure the distortion of $\mathbf{S}$ by

$$\mathrm{err}_{\mathrm{di}}(\tau) = \ln\frac{\lambda_3}{\lambda_1} \tag{7}$$

where $0 < \lambda_1 \leq \lambda_2 \leq \lambda_3$ are the eigenvalues of $\mathbf{S}(\tau)$. A perfectly spherical $\mathbf{S}(\tau)$ gives $\mathrm{err}_{\mathrm{di}} = 0$, and by construction $\mathrm{err}_{\mathrm{di}}$ is invariant to uniform scaling of $\mathbf{S}$.

Finally we define the global error using (6) and (7) as

$$\mathrm{err}(\tau) = \max\{\,\mathrm{err}_{\mathrm{nl}}(\tau),\, \mathrm{err}_{\mathrm{di}}(\tau)\}\;\; \tag{8}$$

3.3. Autonomous particles

We define the state of an autonomous particle starting at time t_S as a tuple

$$P = (\mathbf{x}_s, \mathbf{S}_s, \nabla\phi_s, \mathbf{x}_0, t_0, \mathbf{S}_0, k) \ .$$

The components $\mathbf{x}_s, \mathbf{S}_s$ denote the initial location and ellipsoidal neighborhood when integration started. The components $\mathbf{x}_0, t_0, \mathbf{S}_0$ refer to the last "safe" point during the integration such that

$$\begin{aligned} \mathbf{x}_0 &= \phi(\mathbf{x}_s, t_s, t_0 - t_s) \\ \nabla\phi_s &= \nabla\phi(\mathbf{x}_s, t_s, t_0 - t_s) \\ {\mathbf{S}_0}^2 &= \nabla\phi_s \, {\mathbf{S}_s}^2 \, (\nabla\phi_s)^T \ . \end{aligned}$$

For convenience, we provide a condition for the squared matrix ${\mathbf{S}_0}^2$ as introduced in section 3.1. The matrices $\mathbf{S}_0$ and ${\mathbf{S}_0}^2$ are symmetric positive definite, and we obtain $\mathbf{S}_0$ as the matrix square root, which can be computed, e.g., from the spectral decomposition of ${\mathbf{S}_0}^2$. The component k denotes the current split depth. We track k to limit the number of splits of a particle to a maximum of $k_{\max}$.

Starting from the last "safe" point $(\mathbf{x}_0, t_0, \mathbf{S}_0)$, we advect $\mathbf{x}, \mathbf{S}$ for a period τ and obtain $\nabla\phi(\mathbf{x}_0, t_0, \tau)$, $\mathbf{S}(\tau)$ and $\mathrm{err}(\tau)$ as defined in (4), (5) and (8). We observe the error $\mathrm{err}(\tau)$ during the advection and require that a threshold $\mathrm{err}_{\mathrm{split}}$ is never exceeded. Then one of two cases applies: either, the error stays below the threshold for the entire period until $t_0 + \tau$, or a time t_{split} at which the error exceeds the threshold for the first time. Depending on which case applies, we update the state of the particle as follows:

The error threshold is *not* exceeded: The advection reaches the end time $t_e = t_0 + \tau$ with $\mathrm{err}(\hat{\tau}) < \mathrm{err}_{\mathrm{split}}$ for all $0 \le \hat{\tau} \le \tau$. In this case, the particle state is updated as

$$\begin{pmatrix} \mathbf{x}_s \\ \mathbf{S}_s \\ \nabla\phi_s \\ \mathbf{x}_0 \\ t_0 \\ \mathbf{S}_0 \\ k \end{pmatrix} \rightarrow \begin{pmatrix} \mathbf{x}_s \\ \sqrt{(\nabla\phi\ \nabla\phi_s)^{-1}\, \mathbf{H}\, \mathbf{H}^T\, (\nabla\phi\ \nabla\phi_s)^{-T}} \\ \nabla\phi\ \nabla\phi_s \\ \phi \\ t_e \\ \sqrt{\mathbf{H}\,\mathbf{H}^T} \\ k \end{pmatrix} ,$$

where $\phi = \phi(\mathbf{x}_0, t_0, \tau)$, $\nabla\phi = \nabla\phi(\mathbf{x}_0, t_0, \tau)$, and $\mathbf{H} = \mathbf{H}(\tau)$. In this case, the advection of the autonomous particle stops at time t_e.

The error threshold is exceeded: The advection reaches a time $0 < t_{\mathrm{split}} < t_e$ such that

$$\mathrm{err}(t_{\mathrm{split}} - t_0) > \mathrm{err}_{\mathrm{split}}$$

for the first time, i.e., $\mathrm{err}(t - t_0) \le \mathrm{err}_{\mathrm{split}}$ for all $t_0 \le t < t_{\mathrm{split}}$. In this case, we *split* the particle into three new particles as follows. (Technically, we update one particle and create two new particles.)

1:3 split of a particle. Splitting a particle with state P into three particles gives three state transitions

$$P \ \rightarrow \ P_i' \quad for \quad i = 1,2,3 \ ,$$

which are obtained by refining along the major axis of the ellipsoid neighborhood as follows. Let

$$\mathbf{H}\mathbf{H}^T \ = \ \mathbf{U}\Lambda\mathbf{U}^T$$

denote the spectral decomposition of $\mathbf{H}\mathbf{H}^T$ where the diagonal matrix Λ captures the eigenvalues $0 < \lambda_1 \le \lambda_2 \le \lambda_3$, and the eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ are the columns of the orthogonal matrix $\mathbf{U}$. For

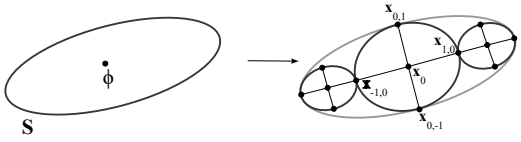

Figure 3: *Split of autonomous particle with linear neighborhood* $\mathbf{S}$ *to three new particles and neighborhoods.*

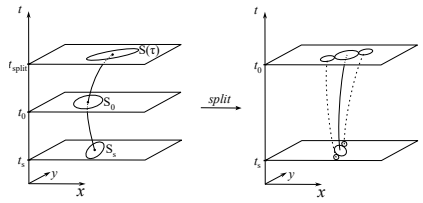

Figure 4: *Split of autonomous particle at time* t_{split} *creates three new autonomous particles.*

$i = 1,2,3$, we define the new states as

$$P_i' \ = \ \begin{pmatrix} \mathbf{x}_s \ + \ \triangle_i^{\mathbf{x}} \\ \sqrt{(\nabla\phi\ \nabla\phi_s)^{-1}\, \mathbf{U}\, \Lambda_i'\, \mathbf{U}^T\, (\nabla\phi\ \nabla\phi_s)^{-T}} \\ \nabla\phi\ \nabla\phi_s \\ \phi \ + \ \triangle_i^{\phi} \\ t_{\mathrm{split}} \\ \mathbf{U}\, \Lambda_i'\, \mathbf{U}^T \\ k+1 \end{pmatrix} ,$$

with displacements

$$\triangle_i^{\mathbf{x}} \ = \ \begin{cases} \mathbf{0} & \text{for } i = 1 \\ \pm\frac{3}{4}\sqrt{\lambda_3}\,(\nabla\phi\nabla\phi_S)^{-1}\,\mathbf{u}_3 & \text{for } i = 2,3 \end{cases}$$

$$\triangle_i^{\phi} \ = \ \begin{cases} \mathbf{0} & \text{for } i = 1 \\ \pm\frac{3}{4}\sqrt{\lambda_3}\mathbf{u}_3 & \text{for } i = 2,3 \end{cases}$$

and nonuniform scaling by diagonal matrices

$$\Lambda_i' \ = \ \begin{cases} \mathrm{diag}(\ \sqrt{\lambda_1},\ \sqrt{\lambda_2},\ \frac{1}{2}\sqrt{\lambda_3}) & \text{for } i = 1 \\ \mathrm{diag}(\frac{1}{2}\sqrt{\lambda_1},\ \frac{1}{2}\sqrt{\lambda_2},\ \frac{1}{4}\sqrt{\lambda_3}) & \text{for } i = 2,3 \end{cases} .$$

Figures 3 and 4 illustrate the splitting process. The local coordinate system for translation and scaling is induced by the eigenvectors $\mathbf{u}_i$. Splitting generates three new ellipsoidal regions with centers positioned along the main axis in direction $\mathbf{u}_3$ with a nonuniform scaling using eigenvalues λ_i. The translation and the scaling are chosen carefully: The three generated ellipsoids are separated by their centers, they do not intersect, they are all located within and cover a large area of the original neighborhood.

3.4. Seeding autonomous particles

To start the integration of autonomous particles, we place a certain number of particles $\mathbf{x} \in \mathcal{S}$ from a set of seeds $\mathcal{S}$. Their initial neighborhoods are spheres of radius $r_{\mathcal{S}}$. We arrange the seeds on a regular grid with a rather coarse cell size and chose the maximum radius $r_{\mathcal{S}}$ such that adjacent neighborhoods touch but do not intersect. To fill the gaps that appear between four adjacent particles (or eight particles in 3D), we add a second set of particles that are arranged

on a grid that is offset by $\frac{r_S}{2}$. For each particle $\mathbf{x} \in \mathcal{S}$, its initial state is $(\mathbf{x}, r_S \mathbf{I}, \mathbf{I}, \mathbf{x}, t_s, r_S \mathbf{I}, 0)$, where all particles start at time t_S.

3.5. Flow map reconstruction

The set of advected and split autonomous particles gives a discrete representation of the flow map. To continuously evaluate the flow map or its inverse at any domain point, we require an interpolation scheme. There are various methods for scattered data interpolation. We prefer a local reconstruction and chose Sibson's natural neighbor interpolation [Sib81], which has been adapted for flow map reconstruction [BT13]. This interpolation scheme is simple because it requires essentially a (assuming normalized weights) barycentric combination of samples in the neighborhood of the evaluation point. We use the CGAL [The22] library.

3.6. Reseeding

Flow maps typically show exponential behavior, i.e., there is strong distortion of linear neighborhoods. This can lead to excessive particle splitting in certain regions, e.g., near FTLE ridges. As a consequence, the number of particles increases exponentially without significant benefit for the reconstruction. We limit the number of splits for each particle to $k_{\max}$. However, not being able to split anymore affects the adaptation. To cope with this problem and to effectively limit the number of particles, we apply temporal reseeding as proposed Agranovsky et al. [ACG*14]: The temporal domain $[t_0, t_0+\tau]$ is partitioned into N time intervals of length $\frac{\tau}{N}$. This result in a piecewise flow map in τ that consists N pieces $\phi_i(\mathbf{x}, t_0 + (i-1)\frac{\tau}{N}, \frac{\tau}{N})$ for $i = 1, \ldots, N$.

Agranovsky et al. [ACG*14] advect particles that are always seeded from a sufficiently dense *rectilinear grid* at start times $t_0 + (i-1)\frac{\tau}{N}$ and advected for a period $\frac{\tau}{N}$. In contrast, we advect autonomous particles: As their advection adapts to separation by splitting, we also start from a regular but can use a less dense grid of seeds. Due to the adaptation, we require particles and fewer steps of numerical integration for a similar reconstruction error.

4. Results

4.1. Double Gyre

As benchmark data set we use the 2D double gyre flow introduced by Shadden [SLM05] with parameters $\varepsilon = \frac{1}{4}$, $\omega = \frac{2\pi}{10}$, $A = \frac{1}{10}$. The vector field consists of two regions that share a vertical boundary. The boundary and the centers of the two eddies are moving periodically from left to right and back. Its domain is defined in $[0,2] \times [0,1]$.

For measuring the accuracy we compare the reconstructed flow map fields $\phi_{\text{ap}}$from our method and $\phi_{\text{agra}}$from [ACG*14]. We construct a ground truth flow map ϕ_{gt} as the advection of massless particles that have been integrated numerically using an adaptive 7th-order Runge-Kutta-Fehlberg scheme and define errors

$$e_{\text{ap}} = ||\phi_{\text{ap}} - \phi_{\text{gt}}||_2^2 \quad \text{and} \quad e_{\text{agra}} = ||\phi_{\text{agra}} - \phi_{\text{gt}}||_2^2 .$$

A negative difference

$$c = e_{\text{ap}} - e_{\text{agra}} \tag{9}$$

indicates that our method performs better than [ACG*14]. We compute the errors and diffrences on a regular grid of a solution of 2000×1000.

We use the following parameters for the advection of the autonomous particles: period $\tau = 0.1$, $|\mathcal{S}| = 20 \cdot 10 + 19 \cdot 9 = 371$, $k_{\max} = 3$. After the advection of the initial set of autonomous particles we have a total number of 57915 particles with a temporal resolution of $N = 5$. To make the comparisons fair we try to match the number of final particles that are being advected with the cumulative number of grid cells of [ACG*14] (see table 2).

Our method					
N_{ap}	1	2	3	4	5
#adv.particles	11,583	23,166	34,749	46,332	57,915
Agranovksy					
N_{agra}	Total number of advected particles				
1	11,858	23,328	23,765	46,818	58,482
2	11,990	23,562	35,156	47,306	58,564
3	12,015	23,625	35,343	47,259	58,806
4	12,012	23,328	34,848	47,124	58,824
5	12,075	23,765	35,700	47,265	58,905

Table 2: *Number of advected Particles in parameter study (see **additionalmaterial**). The numbers of particles used for the method by Agranovsky et al. approximately equals the number of particles of our approach.*

In addition, we interpolate the samples that are generated by [ACG*14] using [BT13]. Figure 5 shows error plots for forward and backward reconstructions. For both methods, ours and [ACG*14], we observer larger errors near FTLE ridges .

The figures in the **additional material** show a parameter study to compare the impact of the temporal resolutions N_{ap} (for our approach) and N_{agra} (for [ACG*14]). Tables 3 and 4 show the averaged errors over all samples for the temporal resolution setups in these figures. N_{ap} grows from left to right and N_{agra} grows from top to bottom.

The top row shows $N_{\text{agra}} = 1$ which means that a standard flow map sampling on a rectilinear grid was computed. For a positive advection time (see **additional material**) our method yields better results for this scenario with $N_{\text{ap}} > 2$ around the FTLE ridges. The average error is smaller for every N_{ap}. For $N_{\text{ap}} = 5$ the average error of our approach is smaller in most cases of $N_{\text{agra}} = 5$.

Our method					
N_{ap}	1	2	3	4	5
error	1.07e-2	6.27e-3	2.86e-3	2.35e-3	2.38e-3
Agranovksy					
N_{agra}	error				
1	1.80e-2	1.13e-2	8.43e-3	7.04e-3	6.15e-2
2	6.79e-3	3.54e-3	2.62e-3	1.89e-3	1.47e-2
3	4.13e-3	1.42e-3	0.51e-3	0.65e-3	0.59e-2
4	3.68e-3	1.72e-3	1.15e-3	0.72e-3	0.63e-2
5	7.25e-3	1.93e-3	1.26e-3	0.95e-3	6.39e-5

Table 3: *Accumulated flow map errors of double gyre with varying N and positive advection time τ. Green: our method performs better. Red: [ACG*14] performs better.*

Our method					
N_{ap}	1	2	3	4	5
error	1.59e-2	6.57e-3	3.49e-3	3.24e-3	1.84e-3
Agranovksy					
N_{agra}	error				
1	1.13e-2	6.44e-3	4.99e-3	4.13e-3	3.48e-3
2	8.40e-3	5.68e-3	4.40e-3	3.57e-3	3.07e-3
3	7.44e-3	4.53e-3	3.42e-3	2.86e-3	2.43e-3
4	7.41e-3	3.99e-3	2.85e-3	2.27e-3	1.84e-3
5	7.25e-3	3.32e-3	2.09e-3	2.78e-3	2.29e-3

Table 4: *Accumulated flow map errors of double gyre with varying N and negative advection time τ. Green: our method performs better. Red: [ACG*14] performs better.*

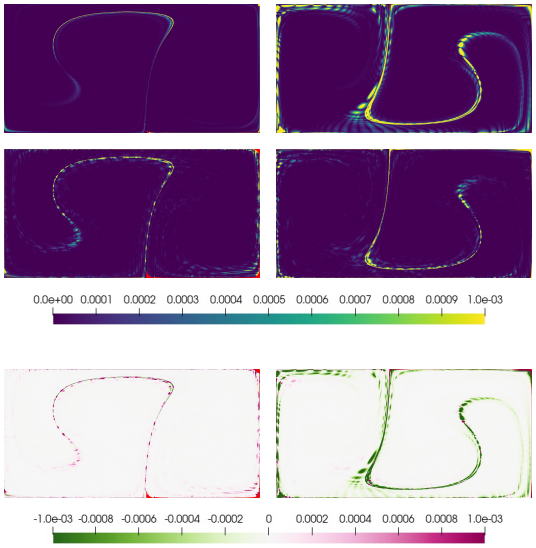

Figure 5: *Error plots of the flow map of the doublegyre for our approach and for [ACG*14]. From top to bottom: e_{ap} | e_{agra} | $c = e_{ap} - e_{agra}$, left $\tau = 10$, right $\tau = -10$*

For a negative advection time our method performs better in most areas for arbitrary N_{ap}. For $N_{ap} > 2$ our approach performs better in nearly every situation. The average error is always smaller for $N_{ap} = 5$ and in certain cases for smaller N_{ap}.

4.2. Channel flow

A fully developed turbulent flow in a long channel has been simulated by direct numerical simulation (DNS) using the DNS solver

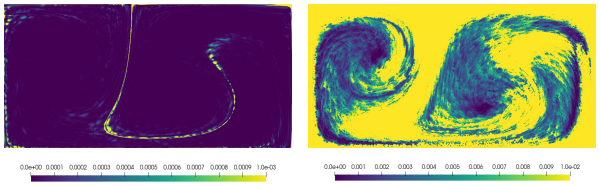

Figure 6: *Reconstruction with with Sibson's (left) and Shepard's method (right). Note that scales differ by a factor of 10.*

Re_τ	Re_b	U_b (m/s)	u_τ (m/s)	l^* (μm)	η (μm)
906	38,432	23.9	1.12	13.8	27.6

Table 5: *Summary of the DNS case. Here, U_b is the bulk velocity, $Re_b = 2hU_b/\nu$ is the bulk Reynolds number, u_τ is the friction velocity, Re_τ is the friction Reynolds number, $l^* = \nu/u_\tau$ is the viscous length scale, and η is the Kolmogorov length scale near the wall.*

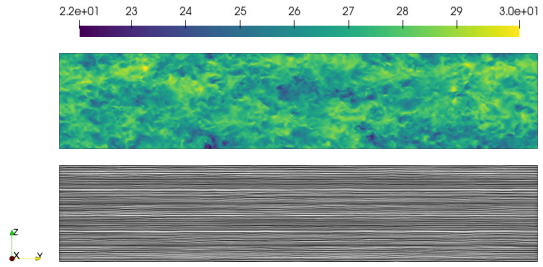

Figure 7: *Top: Velocity magnitude $||\mathbf{v}||$, bottom: Line Integral Convolution [CL93] of the velocity (bottom) of the channel flow.*

DINO [AFO*16, CAT20, OCGT22]. There is 6th-order centered explicit scheme for spatial derivatives and a 3rd-order explicit Runge-Kutta scheme for temporal integration in DINO. The Poisson equation for pressure is solved with spectral accuracy.

The properties of the turbulent channel flow are summarized in Table 5. Here, $Re_\tau = hu_\tau/\nu$, where h is the half-width of the channel and $\nu = 1.557 \times 10^{-5}$ m^2/s. The channel domain has length $20h \times 2h \times 4h$ with $h = 0.0125$ m. This domain is discretized by $1024 \times 1025 \times 256$ grids, with uniform spacing in the streamwise and spanwise directions and stretched spacing in the wall-normal direction. We use u, v, and w to denote streamwise, wall-normal, and spanwise velocity fluctuations, respectively, corresponding to the x-, y-, and z-directions. The final spatial resolution is $\Delta x^+ = 17.56$, $\Delta z^+ = 14.05$, and $0.35 \leq \Delta y^+ \leq 8.40$, which is sufficient to capture the small scale turbulences. The superscript $+$ refers to quantities normalized by the friction velocity u_τ and the viscous wall unit $l^* = \nu/u_\tau$.

We advected a set $\mathcal{S}$ of initial Autonomous Particles in the domain of the channel flow with a resolution of $30 \cdot 300 \cdot 60 + 29 \cdot 299 \cdot 59 = 1\,051\,589$ and a $k_{max} = 3$. Since the channel flow has a

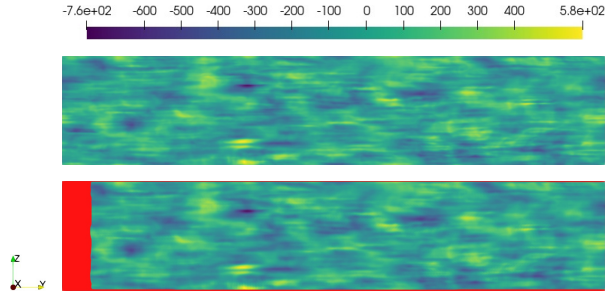

Figure 8: *x-coordinate of flow map. Top: numerical integration, bottom: reconstruction with autonomous particles. Only the x-coordinate is color coded as y and z evolve rather constant and show a simple gradient.*

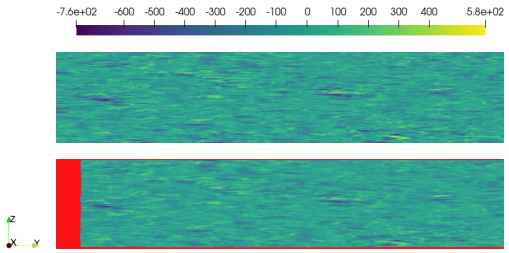

Figure 9: *FTLE field. Top: numerical integration, bottom: reconstruction with autonomous particles (red: no reconstruction).*

very high velocity, only a small advection time can be considered. We used $\tau = 5 \times 10^{-4}$, $N = 10$. The period τ of the Autonomous Particles is set to 5×10^{-6} The sampling of the error e_{ap} and e_{agraA} and c are computed on regular grid with $200 \times 2000 \times 400$ nodes.

For a better view of the data we are showing a 2D slice of the 3D domain in the center of the x-range.

Fig. 7 shows the velocity magnitude and a LIC image [CL93]. Even if the LIC image of this slice seems to be laminar the full 3D flow is turbulent as can be seen in the velocity magnitude $||\mathbf{v}||$. Fig. 8 shows the x-component of the ground truth flow map and the flow map created by our method. Besides minor artifacts both fields are visually identical. The same is true for the ground truth FTLE field and the FTLE field that was computed with the flow map that was created by our method (fig. 9).

In figures 8 and 9, red encodes regions where no flow map data is available. This is the case because the initial set $\mathcal{S}$ is advected in positive y-direction which leads to areas with no autonomous particles near $y = 0$. Sibson's interpolation scheme [Sib81] results in these areas because interpolation is only possible in the convex hull of the scattered data.

5. Discussion

We present an approach to flow map sampling and reconstruction that fulfills our design goals (1.)–(5.) as postulated in section 1. In particular, our approach is adaptive and suited for in-situ sampling, e.g., during a simulation, or for streaming data, e.g., from mass storage: We strictly proceed forward in time, and the implementation requires minimal interference with the host application, because we only require evaluation of the flow field.

Reconstruction. Flow map reconstruction is a scattered data interpolation problem. There are various interpolation methods. We excluded global methods like radial basis interpolation, because all design choices for autonomous particles strive to be local or independent. Also, the benefit of a global interpolation method is unclear: Concerning cost, we need to solve at least a (possibly large) linear system. Concerning accuracy, there may be no benefit due to the (possibly) large flow map gradients.

The simplest local method is probably Shepard's interpolation, e.g., by inverse distance weighting. It turns out that this method may be too simple. More importantly, it requires a global parameter: the radius defining the local neighborhood. Our experiment showed that firstly, it is difficult to find a good parameter setting. Secondly, the achieved reconstruction error is significantly higher than for natural neighbor interpolation. A similar argument applies to moving least-squares approximation. In contrast, Sibson's interpolation [Sib81, BT13] is parameter-free, and we see this as a significant advantage.

Performance. The computations of the tests for the double gyre have been done on a machine with an Intel Core i7-7700K CPU with 4.20GHz and 8 virtual cores. The advection of the particles for the double gyre flow takes about 5 minutes. A reconstruction at 2000×1000 sample points takes about 2 minutes.

The computations of the turbulent channel flow have been done on a machine with 50 cores of a Intel Xeon E5-2690 each with 2,6 GHz that are provided by a virtual machine on a server. This machine has 300GB of RAM. The advection of the autonomous particles in the turbulent channel flow took about 4 hours. A reconstruction of the flow map with $200 \times 2000 \times 400$ samples takes about 10 minutes.

Application scenarios. We constructed our method to be *in-situ friendly*, i.e., it can run in a possibly large simulation running on multiple nodes with low overhead in terms of computational cost, communication and interfacing for the implementation. In this setting, only a small fraction of the simulated data is stored, i.e., the flow map sampling must proceed with the simulation. This is, however, only one possible scenario.

Similarly important are scenarios, where all data, e.g., all timesteps of a simulated flow, is available on external storage. If the data is huge, reading, e.g., from storage arrays or networks, is expensive and reading the entire data set into memory is impossible. The only option may be *streaming* the data and processing only within a small temporal window. In fact, this is the scenario for our experiments. The size of the turbulent channel flow is 8GB.

Evaluation times. We present our method for evaluation of the flow map after a fixed period τ. For the experiments, this is reasonable because we can expect higher reconstruction error for higher integration times. However, an extension to evaluate for arbitrary integration times $\tau' \in [0, \tau]$ may be useful in practice. Such an extension is possible, however, it may require some extra considerations like "reusing" or incrementally updating Voronoi partitions for natural neighbor interpolation.

Temporal reseeding. We adopt the temporal reseeding from Agranovsky et al. [ACG*14] to limit the number of autonomous particles. This is necessary due to the exponential behavior of flow maps, which leads to locally excessive splitting. In our current method, particles can only split. There is nothing like merge events: they are impossible without neighborhood tracking, which would not only complicate the method itself but would also require significantly more interdependence with, e.g., a simulation. One possible direction may be an extension to probabilistic sampling with a criterion for discarding particles (or splits) based on a a random process that can be evaluated autonomously. We leave this, however, as subject of future work.

Acknowledgments

This work is supported by the Priority Programme SPP 1881 *Turbulent Superstructures* of the Deutsche Forschungsgemeinschaft (DFG), grant TH 692/18-1 and TH 881/30-1.

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 *LDAV* (2014), IEEE Computer Society, pp. 67–75. 2, 3, 6, 7, 8

[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. *Comput. Fluids 131* (2016), 123–141. 7

[AOGJ15] AGRANOVSKY A., OBERMAIER H., GARTH C., JOY K. I.: A multi-resolution interpolation scheme for pathline based lagrangian flow representations. In *Visualization and Data Analysis* (2015), vol. 9397 of *SPIE Proceedings*, SPIE, p. 93970K. 3

[BJ15] BUJACK R., JOY K. I.: Lagrangian representations of flow fields with parameter curves. In *5th IEEE Symposium on Large Data Analysis and Visualization, LDAV 2015, Chicago, IL, USA, October 25-26, 2015* (2015), Bennett J., Childs H., Hadwiger M., (Eds.), IEEE Computer Society, pp. 41–48. 3

[BT13] BARAKAT S. S., TRICOCHE X.: Adaptive refinement of the flow map using sparse samples. *IEEE transactions on visualization and computer graphics 19*, 12 (2013), 2753–2762. 2, 3, 6, 8

[CAT20] CHI C., ABDELSAMIE A., THÉVENIN D.: A directional ghost-cell immersed boundary method for incompressible flows. *Journal of Computational Physics 404* (2020), 109122. 7

[CBJ16] CHANDLER J., BUJACK R., JOY K. I.: Analysis of Error in Interpolation-Based Pathline Tracing. In *EuroVis 2016 - Short Papers* (2016), Bertini E., Elmqvist N., Wischgoll T., (Eds.), The Eurographics Association. 3

[CL93] CABRAL B., LEEDOM L. C.: Imaging vector fields using line integral convolution. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1993), SIGGRAPH '93, Association for Computing Machinery, p. 263–270. 7, 8

[COJ15] CHANDLER J., OBERMAIER H., JOY K. I.: Interpolation-based pathline tracing in particle-based flow visualization. *IEEE Transactions on Visualization and Computer Graphics 21*, 1 (2015), 68–80. 3

[ELPH19] ENGELKE W., LAWONN K., PREIM B., HOTZ I.: Autonomous particles for interactive flow visualization. *Comput. Graph. Forum 38*, 1 (2019), 248–259. 3

[GGTH07] GARTH C., GERHARDT F., TRICOCHE X., HANS H.: Efficient computation and visualization of coherent structures in fluid flow applications. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1464–1471. 2, 3

[GHCW21] GU P., HAN J., CHEN D. Z., WANG C.: Reconstructing unsteady flow data from representative streamlines via diffusion and deep-learning-based denoising. *IEEE Computer Graphics and Applications 41*, 6 (2021), 111–121. 3

[GYH*20] GUO L., YE S., HAN J., ZHENG H., GAO H., CHEN D. Z., WANG J.-X., WANG C.: Ssr-vfd: Spatial super-resolution for vector field data analysis and visualization. In *2020 IEEE Pacific Visualization Symposium (PacificVis)* (2020), pp. 71–80. 3

[HBJG16] HUMMEL M., BUJACK R., JOY K. I., GARTH C.: Error estimates for lagrangian flow field representations. In *Eurographics Conference on Visualization, EuroVis 2016, Short Papers, Groningen, The Netherlands, 6-10 June 2016* (2016), Bertini E., Elmqvist N., Wischgoll T., (Eds.), Eurographics Association, pp. 7–11. 3

[HSW11] HLAWATSCH M., SADLO F., WEISKOPF D.: Hierarchical line integration. *IEEE Trans. Vis. Comput. Graph. 17*, 8 (2011), 1148–1163. 2, 3

[JGG21] JAKOB J., GROSS M., GÜNTHER T.: A fluid flow data set for machine learning and its application to neural flow map interpolation. *IEEE Transactions on Visualization and Computer Graphics 27*, 2 (2021), 1279–1289. 3

[KER*14] KUHN A., ENGELKE W., RÖSSL C., HADWIGER M., THEISEL H.: Time line cell tracking for the approximation of lagrangian coherent structures with subgrid accuracy. *Computer Graphics Forum 33*, 1 (2014), 222–234. 2, 3

[OAM*18] OSTER T., ABDELSAMIE A., MOTEJAT M., GERRITS T., RÖSSL C., THÉVENIN D., THEISEL H.: On-the-fly tracking of flame surfaces for the visual analysis of combustion processes. *Computer Graphics Forum 37*, 6 (2018), 358–369. 3

[OCGT22] OU Z., CHI C., GUO L., THÉVENIN D.: A directional ghost-cell immersed boundary method for low mach number reacting flows with interphase heat and mass transfer. *Journal of Computational Physics 468* (2022), 111447. 7

[RPD20] RAPP T., PETERS C., DACHSBACHER C.: Void-and-cluster sampling of large scattered data and trajectories. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (2020), 780–789. 2, 3

[SB21] SAHOO S., BERGER M.: Integration-Aware Vector Field Super Resolution. In *EuroVis 2021 - Short Papers* (2021), Agus M., Garth C., Kerren A., (Eds.), The Eurographics Association. 3

[Sib81] SIBSON R.: A brief description of natural neighbor interpolation. In *Interpreting Multivariate Data* (1981), Barnet V., (Ed.), pp. 21–36. 3, 6, 8

[SLM05] SHADDEN S. C., LEKIEN F., MARSDEN J. E.: Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. *Physica D 212*, 7 (2005). 6

[SP07] SADLO F., PEIKERT R.: Efficient visualization of lagrangian coherent structures by filtered amr ridge extraction. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1456–1463. 2, 3

[The22] THE CGAL PROJECT: *CGAL User and Reference Manual*, 5.5.1 ed. CGAL Editorial Board, 2022. 3, 6