Predictor-Corrector Schemes for Visualization of Smoothed Particle Hydrodynamics Data

Benjamin Schindler, Raphael Fuchs, John Biddiscombe, and Ronald Peikert, *Member, IEEE*

Abstract—In this paper we present a method for vortex core line extraction which operates directly on the smoothed particle hydrodynamics (SPH) representation and, by this, generates smoother and more (spatially and temporally) coherent results in an efficient way. The underlying predictor-corrector scheme is general enough to be applied to other line-type features and it is extendable to the extraction of surfaces such as isosurfaces or Lagrangian coherent structures. The proposed method exploits temporal coherence to speed up computation for subsequent time steps. We show how the predictor-corrector formulation can be specialized for several variants of vortex core line definitions including two recent unsteady extensions, and we contribute a theoretical and practical comparison of these. In particular, we reveal a close relation between unsteady extensions of Fuchs et al. and Weinkauf et al. and we give a proof of the Galilean invariance of the latter.

When visualizing SPH data, there is the possibility to use the same interpolation method for visualization as has been used for the simulation. This is different from the case of finite volume simulation results, where it is not possible to recover from the results the spatial interpolation that was used during the simulation. Such data are typically interpolated using the basic trilinear interpolant, and if smoothness is required, some artificial processing is added. In SPH data, however, the smoothing kernels are specified from the simulation, and they provide an exact and smooth interpolation of data or gradients at arbitrary points in the domain.

Index Terms—Smoothed particle hydrodynamics, flow visualization, unsteady flow, feature extraction, vortex core lines.

1 Introduction

The smoothed particle hydrodynamics (SPH) method was invented in the 1970's for astrophysical simulations. Recently, SPH has become an alternative to traditional Eulerian methods also in industrial computational fluid dynamics (CFD) applications. It is especially useful in problems where a free surface is present or in multi-phase flow simulations. Here, strong initiatives are being taken by academia and industry to make SPH a reliable tool to be used for practical engineering problems. The reason why SPH simulations have not been regularly used much earlier in industrial CFD lies in the fact that SPH was originally designed for unbounded domains. Much of the current research on the numerics side has to do with the modeling of boundaries [11].

Two obvious approaches to visualize SPH data would be resampling and triangulation. Resampled data, especially on uniform grids, can be post-processed with a broad range of visualization algorithms and implementations in commercial or academic software packages. There are, however, some major drawbacks linked to this approach. When resampling data of varying sampling density, either the density must match the highest density, causing an increase in data size, or detail is lost in some regions. Also, solid walls and free surfaces are difficult to represent in structured grids. And, finally, resampling causes the dilemma of either blurring data or generating artifacts. The alternative, to triangulate point-sampled data, is very time-consuming. A triangulation functionality is not often found in visualization software, and it has difficulties with concavities in domain boundaries. Finally, both resampling and triangulation induce an interpolation function which is not compatible with SPH interpolation.

There exist visualization algorithms and software that are capable of operating directly on SPH data. However, most of these are either limited to basic visualization techniques or they treat SPH data as general point-sampled data, this way not fully exploiting the information contained in the SPH representation.

Our approach is to respect the kernel functions that were used for the simulation and to reuse them also for visualization. The advantage is that at arbitrary points in the domain all interpolated values, including derivatives, are exact with respect to the SPH representation. However, it implies that visualization algorithms relying on the cell structure of grids cannot be used but must be adapted or replaced by new algorithms. Compared to other data representations, SPH data allows us to compute gradients and second derivatives more efficiently and in higher quality. This fact can be exploited especially in visualization problems that can be tackled with a predictor-corrector strategy. This holds for the extraction of line-type features, the problem addressed in this paper, but also for other types of features or other geometric objects such as isosurfaces. Both prediction and correction steps make heavy use of derivatives, therefore this strategy is appropriate for SPH.

Some classes of grid based visualization rely on the connectivity of a grid in the sense that they compute in a cell-by-cell manner. Examples are Marching Cubes type contouring methods and parallel vectors methods. Such methods can be reformulated as (spatial) tracking methods if it is possible to estimate the tangent of the feature curve or the tangential plane of the feature surface. Error accumulation in such tracking methods can be avoided by including correction steps, resulting in *predictor-corrector schemes*. These have the advantage that they work without a grid, but the price to pay is that derivatives are needed for the tangent estimation. In principle, a numerical integration method for stiff ODEs could be used instead of the correction steps. But for problems such as isosurfacing or finding parallel vectors, the error can be detected locally and correction to an exact feature point is possible. This is different from the standard case of streamline integration where a good integration method is the only choice to reduce errors. The simplicity of the prediction and correction steps makes this approach attractive from a performance point of view.

The properties of predictor-corrector schemes make them ideally suited for SPH visualization where no grid is available but where derivatives can be fast and accurately computed. Moreover, in time-dependent data, prediction can be made for the feature in the subsequent time step. This way, temporal coherence is exploited. The efficiency of spatial tracking methods depends on the seeding strategy. Computing seed points should be fast, enough seed points must be generated not to miss a feature, and redundancy should be minimized. An important factor is the availability of a *fast rejection test* which helps to quickly traverse space where there is no feature.

- *Benjamin Schindler, Raphael Fuchs, and Ronald Peikert are with Institute of Visual Computing, ETH Zurich, Switzerland, E-mail: {bschindler, raphael, peikert}@inf.ethz.ch.*
- *John Biddiscombe is with Swiss National Supercomputing Centre, Manno, Switzerland, E-mail: biddisco@cscs.ch.*

Manuscript received 31 March 2009; accepted 27 July 2009; posted online 11 October 2009; mailed on 5 October 2009.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org .

In this paper we focus on vortex core lines, although the presented methods can be applied to more general line-type features as long as they can be expressed with the parallel vectors operator [15]. Vortex core lines are a useful representation of vortices which, especially in unsteady flow, cannot be visually inferred from a streamline or pathline pattern. We present an algorithm to extract vortex core lines directly from SPH data, and we detail out versions for the vortex core line criteria defined by Levy et al. [9], Sujudi and Haimes [23], Weinkauf et al. [27], and Fuchs et al. [4]. As a second contribution, we compare the latter two, which extend the Sujudi-Haimes criterion to unsteady flow, with the older criteria that implicitly assume steady flow. A comparison will be done on the level of the definitions, resulting in an equivalent formulation of Weinkauf et al.'s criterion in terms of the acceleration vector which reveals the Galilean invariance of this criterion. Based on two time-dependent data sets, we compare results of the mentioned variants, and we compare results of direct SPH visualization with those of visualization based on resampled data.

2 Background and Related Work

2.1 Smoothed Particle Hydrodynamics

SPH was introduced by Gingold and Monaghan [5] and Lucy [10] for astrophysical problems, and has later become a general CFD method. A review of SPH theory and application can be found in [13]. Here, we focus on aspects of SPH that are most relevant for visualization.

The *interpolation* rule for a quantity A is

$$A(\mathbf{x}) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} A_j W(\mathbf{x}-\mathbf{x}_j, h_j) \tag{1}$$

where m_j is the mass of the j-th particle, ρ_j its density, A_j the value of A associated with that particle, $\mathbf{x}_j$ its position, h_j its smoothing length, and W is the radially symmetric kernel. The summation is done over all particles having the point $\mathbf{x}$ within their kernel support (cf. [17], equation 3). The *normalized interpolation* is obtained by dividing (2.1) by the interpolation of unity which is

$$\sum_{j=1}^{N} \frac{m_j}{\rho_j} W(\mathbf{x}-\mathbf{x}_j, h_j) \approx 1 \tag{2}$$

The kernel most often used is the *cubic spline*

$$W(r,h) = \frac{1}{\pi h^3} \begin{cases} 1 - \frac{3}{2}q^2 + \frac{3}{4}q^3 & 0 \le q \le 1 \\ \frac{1}{4}(2-q)^3 & 1 \le q \le 2 \\ 0 & q \ge 2 \end{cases} \tag{3}$$

where $\mathrm{q} = \|r\|/\mathrm{h}$. To avoid round-off errors, the interpolation formula is often rewritten as

$$A(\mathbf{x}) = \sum_{j=1}^{N} w_j A_j \mathscr{W}(\mathbf{x}-\mathbf{x}_j, h_j) \tag{4}$$

where $w_j = h_j^{-3} m_j/\rho_j$ and $\mathscr{W}(\mathbf{x}-\mathbf{x}_j, h_j) = h_j^3 W(\mathbf{x}-\mathbf{x}_j, h_j)$ (cf. [17], equation 6). For simplicity, we use the abbreviation $W_j(\mathbf{x})$ for the full interpolation weight of A_j, namely $\frac{m_j}{\rho_j} W(\mathbf{x}-\mathbf{x}_j, h_j)$. With this notation, the normalized interpolation is

$$A(\mathbf{x}) = \sum_{j=1}^{N} A_j W_j(\mathbf{x}) \Bigg/ \sum_{j=1}^{N} W_j(\mathbf{x}) \tag{5}$$

and, using the quotient rule, the gradient can be calculated as

$$\nabla A(\mathbf{x}) = \sum_{j=1}^{N} \big(A_j - A(\mathbf{x})\big) \nabla W_j(\mathbf{x}) \Bigg/ \sum_{j=1}^{N} W_j(\mathbf{x}), \tag{6}$$

and, if needed, the Hessian can be computed along the same lines.

The SPH method introduces a smoothing kernel which imposes a coherency in space and time commensurate with the underlying physics used to define the model. An ideal simulation will produce a field which is as smooth as the real conditions, but a bad model, or a set of results which miss time steps, will introduce unrealistic fluctuations in the fields. It is beyond the scope of this paper which concerns visualization to address the deficiencies in the data itself, though the method used to track vortices tries to be as accurate as the data allows.

2.2 Direct visualization of SPH data

One of the best known visualization packages for SPH data is SPLASH [17], which is capable of producing 2D plots of data by projecting particles onto a plane, and 3D plots by integrating the kernel contributions of all particles intersecting a ray through the view pixel. Additionally, SPLASH can produce a form of surface plot by using an "optically thick" integration of particle contributions where the density of material through which the ray passes forms the absorption term of the transport equation. Both 3D plots provided by SPLASH can be considered as image-space based renderings and whilst a simple 2D streamline plot is possible, there is no true support for object-space feature based generation of images.

Extensions to ParaView [6] given in [2] make it possible to resample SPH data onto planes, grids and arbitrary geometric meshes. Rosenthal et al. presented an isosurfacing algorithm for SPH data and applied it to astrophysical data [21]. It generates *surfels* (surface elements, i.e. points with radius, normal and color information) between selected pairs of neighboring particles. The method can be applied to any point-sampled data, not just SPH data. However, since it is ignores the SPH kernels, the obtained isosurface is not strictly correct in the sense of the SPH model. An alternative algorithm was presented later [20] where the isosurface is computed by solving a PDE. Again it targets general point-sampled data, but it seems possible to use the same strategy to compute an isosurface of a quantity given in SPH representation.

2.3 Vortex core line extraction

Vortices in flow fields can be represented by their axes, also known as vortex core lines. Like for the vortices themselves, there is no unique definition of a vortex core line, but rather a number of criteria defined by several authors. Some definitions include the pressure field [1, 12], but others only make use of the velocity information. Many of these definitions can be expressed in part by the parallel vectors operator [15]. For this purpose, two vector fields $\mathbf{v}$ and $\mathbf{w}$ (which can be original or derived data) are specified. Then, vortex core lines according to this definition are obtained by finding locations of parallel vectors and applying a post-filtering. Some possible choices for the vector fields $\mathbf{v}$ and $\mathbf{w}$ are listed in Table 1.

For extracting the raw feature lines, i.e., curves in space which point-wise fulfill the parallel vectors condition, a number of algorithms have been proposed. Peikert and Roth [15] focused on cell-wise methods which solve for the intersection points with the faces of a cell. A different strategy is to track the feature line, starting from a seed point on the feature. Theisel et al. [25] and Sukharev et al. [24] formulated expressions for the tangent of a feature line, and based on this, algorithms for computing raw feature lines by numeric integration. Van Gelder and Pang pointed out that both these methods are prone to error accumulation. They came up with a method [26] that alternates prediction steps with correction steps.

Post-filtering removes parts of raw features that do not fulfill additional criteria required by the respective feature definition. For instance, the vortex core line definition by Sujudi and Haimes [23] requires that the velocity gradient has a pair of complex conjugate eigenvalues in addition to the above stated parallelism. Alternatively, raw feature points can be tested for a minimum amount of swirl. Furthermore, to remove false positives, it is also advisable to restrict the angle between the velocity and the feature tangent [16] (e.g. to less than 45°). Finally, after removing raw feature points together with incident line segments, the remaining set of feature lines can be checked for a minimum length. Too short lines are considered as noise and removed.

3 A Predictor-Corrector Method for Vortex Core Line Extraction

Two vector fields $\mathbf{v}$ and $\mathbf{w}$ are parallel where their cross product $\mathbf{c} = (c_1, c_2, c_3) = \mathbf{v} \times \mathbf{w}$ vanishes. In the general case, solutions of this equation are lines, not isolated points. Therefore, the zero isosurfaces

v		**w**		Method
$\mathbf{u}$	velocity	$\nabla\times\mathbf{u}$	vorticity	Levy et al. [9]
$\mathbf{u}$	velocity	$\mathbf{e_r}(\nabla\mathbf{u})$	single real eigenvector of velocity gradient	Sujudi and Haimes [23]
$\mathbf{u}$	velocity	$(\nabla\mathbf{u})\mathbf{u}$	steady acceleration	–, eigenvector-free reformulation
$\mathbf{u}$	velocity	$\mathbf{a}$	acceleration $(\nabla\mathbf{u})\mathbf{u}+\partial\mathbf{u}/\partial t$	Fuchs et al. [4]
$\mathbf{a}$	acceleration	$\mathbf{e_r}(\nabla\mathbf{u})$	single real eigenvector of velocity gradient	Weinkauf et al. [27]
$\mathbf{a}$	acceleration	$(\nabla\mathbf{u})\mathbf{a}$	velocity gradient times acceleration	–, eigenvector-free reformulation
∇p	pressure gradient	$(\nabla\nabla p)\nabla p$	pressure Hessian times gradient	Miura and Kida [12]

Table 1. Choices of vectors v and w for vortex core line extraction

of c_1, c_2 and c_3 intersect in a line, as is shown in Fig. 1. The tangent direction of this line is perpendicular to all three gradients ∇c_i. From this we can derive procedures for predicting and for correcting feature points (subsections 3.2 and 3.3). In comparison with Van Gelder and Pang's method [26], our proposed method is restricted to 3-space and does not support inhomogeneous systems of equations. This restriction allows us to formulate a very simple correction step, which in a similar way could also be derived for other feature extraction problems such as isosurfaces or Lagrangian coherent structures.

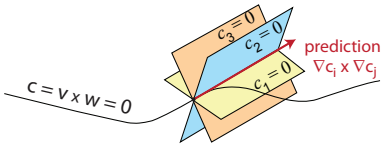

Fig. 1. Zero isosurfaces of components of **c** used for prediction and correction steps.

3.1 Generating seed candidates

In the first stage of the algorithm we need to find potential seed points for our predictor-corrector method. Since this process needs to look at the entire data volume, a very fast rejection test is needed for the algorithm to have reasonable performance.

For the cross product **c** to vanish in the neighborhood of a particle, there must be a zero crossing in every component of **c** and thus both signs must occur for every component of **c** within this neighborhood. To make this test reasonably fast, the radius of the neighborhood is chosen smaller than the kernel support, but still large enough to guarantee a dense covering ($1.5h$ in the case of the cubic spline kernel). Also, these calculations are carried out on raw feature data to avoid expensive SPH interpolations.

As a next step, candidates in regions with no swirl are discarded. This equals to calculating the characteristic polynomial χ of the velocity gradient and only selecting particles where χ has a pair of conjugate complex roots. Instead of just requiring swirl, a minimum *vortex strength* can be prescribed. Vortex strength ω is defined as the imaginary part of the complex eigenvalues of the velocity gradient. For rigid rotation, this is the angular speed, but if a deviatoric strain is added, $|\omega|$ gets smaller, and for pure shear it is zero. Thresholding by vortex strength is especially useful if large portions of the flow are nearly at rest (cf. the example in Section 5.3), because it effectively suppresses small features caused by noise. With the remaining seed candidates, an initial correction step is made before starting the predictor-corrector loop to make sure that the seed candidate lies on a feature line.

3.2 The prediction step

To predict the next feature point from a given position, we need the gradient of the components of $\mathbf{c}=(c_1,c_2,c_3)=\mathbf{v}\times\mathbf{w}$. The gradient of c_i can be expressed by **v**, **w**, and their gradients as follows (with indices taken modulo 3):

$$\nabla c_i = v_{i+1}\nabla w_{i+2} + w_{i+2}\nabla v_{i+1} - v_{i+2}\nabla w_{i+1} - w_{i+1}\nabla v_{i+2} \quad (7)$$

From these three vectors, we compute the pairwise cross products $\nabla c_i\times\nabla c_j$ and select the pair (i,j) which maximizes the magnitude of this cross product. The motivation for this choice is that we want to simultaneously avoid small angles between isosurfaces ($c_i=0$) and small rates of change perpendicular to an isosurface. The direction of this cross product is now taken as the tangent direction of the feature line, and its sign is chosen consistent with the current tracking direction (see Fig. 1). Along this tangent, the next feature point is predicted at a given step size.

3.3 The correction step

Again the gradients ∇c_i are computed using (7), and the pair (i,j) is chosen for which $\nabla c_i\times\nabla c_j$ has maximal magnitude. The idea is now to find a feature point in the plane through the given point **x** and spanned by the two vectors ∇c_i and ∇c_j, that is a point $\mathbf{x}'=\mathbf{x}+s\nabla c_i+t\nabla c_j$. At the point $\mathbf{x}'$ the approximation

$$c_k' \approx c_k + (s\nabla c_i + t\nabla c_j)\cdot\nabla c_k \quad (k=1,\cdots,3) \quad (8)$$

holds. Setting the right hand side to zero for $k=i,j$ leads to the system

$$\begin{aligned} s\nabla c_i\cdot\nabla c_i + t\nabla c_j\cdot\nabla c_i &= -c_i \\ s\nabla c_i\cdot\nabla c_j + t\nabla c_j\cdot\nabla c_j &= -c_j, \end{aligned} \quad (9)$$

for the unknowns s and t. This system is solved for s and t which then yield the corrected point $\mathbf{x}'$. In this approach, we ignore the third component of the cross product, which is fine because the three components fulfill

$$\mathbf{v}\cdot(\mathbf{v}\times\mathbf{w}) = v_1c_1 + v_2c_2 + v_3c_3 = 0. \quad (10)$$

Therefore, when c_1 and c_2 are zero, c_3 becomes also zero as long as $v_3\neq 0$. If $v_3=0$, the same argument can be repeated with **v** replaced by **w** in (10). Only in the degenerate case where $v_3=w_3=0$ can c_3 become nonzero. However, this will be detected if the corrector does not converge to a solution.

The correction step is essentially a Newton-Raphson step, and therefore it can be repeated. We found empirically that two iterations are sufficient to make the error negligible, for prediction step sizes chosen to meet rendering requirements. In case of no convergence, the prediction step can be redone with half the step size, or ultimately, the feature is terminated.

3.4 Usage of given velocity gradients

Typically, at least one of the vectors **v** and **w** involves the velocity gradient. But often the velocity gradients ∇u_i $(i=1,2,3)$ are available from the simulation as particle attributes $\mathbf{g}^i=(g_1^i,g_2^i,g_3^i)$. In this case, (7) can be computed with only first derivatives of the given quantities.

- For the Levy criterion, we have vector components

$$v_i = u_i \quad w_i = g_{i+2}^{i+1} - g_{i+1}^{i+2} \quad (11)$$

and gradients

$$\nabla v_i = \nabla u_i \quad \nabla w_i = \nabla g_{i+2}^{i+1} - \nabla g_{i+1}^{i+2} \quad (12)$$

- For the Sujudi-Haimes criterion, we have $\mathbf{v}=\mathbf{u}$ as above and

$$w_i = \mathbf{g}^i\cdot\mathbf{u} \quad (13)$$

and its gradient is computed using the product rule

$$\nabla w_i = \left(\nabla \mathbf{g}^i\right)\mathbf{u} + (\nabla \mathbf{u})\,\mathbf{g}^i. \tag{14}$$

- For the criterion of Fuchs et al. [4], we have

$$w_i = \mathbf{g}^i \cdot \mathbf{u} + \frac{\partial \mathbf{u}}{\partial t} \tag{15}$$

and its gradient is computed in analogy to (14). Alternatively, because here $\mathbf{w}$ is the particle acceleration, this can be computed more simply and efficiently from particle velocities at two time steps (dividing their difference by the time difference).

- For the criterion of Weinkauf et al. [27], we have

$$v_i = a_i \quad w_i = \mathbf{g}^i \cdot \mathbf{a} \tag{16}$$

($\mathbf{a}$ is acceleration) and gradients computed in analogy to (14).

Note that in (12) and (14) the terms ∇u_i must be computed by using the gradient of the SPH kernels, not using the data attributes $\mathbf{g}^i$ (even if they are available). The reason is that the given gradient data may differ from gradients obtained using kernel gradients. Therefore, to make the predictor-corrector scheme work, the vectors $\mathbf{g}^i$ and $\mathbf{u}$ must be treated as independent quantities.

3.5 Termination criteria

The predictor-corrector process terminates when the boundary of the fluid volume is reached, when the correction step does not converge or when the filtering criteria fail for N consecutive sample points. It also terminates when getting close to another feature line or when encountering a loop (see below).

The way candidate selection works, it is very natural for the candidates to form a cloud around a feature line, thus candidates must be eliminated on the fly to prevent a feature line from being traced multiple times. Since the candidate list not only consists of positions, but also of `ids`, filtering candidates can simply be done by removing any `id` from the candidate list which is inside the neighborhood of a given tracing point. Since the neighborhood is already calculated for SPH interpolation, this step does not cause much overhead. Unfortunately, since candidate points are selected based on raw particle data, this process is not completely reliable. To detect a feature line being drawn multiple times, we save for every particle which core line `id` and step number visited it. Thus, when all neighboring particles at a certain step have already been visited by another core line, the tracing process is terminated. When all neighbors have been visited by the same core line already, there is the possibility of a looping core line.

If we would terminate at this point, looping core lines would not be closed. This is because neighborhoods overlap and the termination would come too early. Instead, we now calculate a similarity criterion loosely inspired by [3]. We first find the point on the core line closest to the last point. We then calculate the average distance of the last N points to the core line. If this distance is smaller than the step size used for the prediction step, we terminate.

3.6 Post-filtering of raw features

In the post-filtering stage, the set of raw features is reduced to the final, meaningful features. For vortex core lines, filtering criteria include the following:

- a pair of complex conjugate eigenvalues of $\nabla \mathbf{u}$
- sufficient vortex strength (see Section 3.1)
- small angle between velocity and the core line.

The small angle criterion is useful, because some of the criteria assume a "swirl" type vortex and do not work well for "tumble" vortices [8]. It is very usual for a criterion to fail along a core line. To get nicely connected core lines, we use a *tolerant* filtering which accepts N consecutive failures of any filtering criterion before aborting. If N is reached, the last N inserted feature points are removed to have a consistent output. In the end, only feature lines longer than a specified limit are accepted.

3.7 Temporal coherence

When creating animated core lines, feature points of the previous time step can be reused. Since this guarantees candidate points for the features already available in the previous time step, we can reduce the amount of searching done when generating seed candidates. In our implementation, we chose to run the candidate test only on every tenth particle, which results in substantial speed-ups (see Fig. 5). This optimization can theoretically lead to a feature being missed for nine consecutive time steps, but in practice almost all features can be traced from many possible candidate points. Comparisons with full candidate searches revealed that only a few very small and short-lived features were missed.

3.8 Higher-order temporal interpolation

To save storage space, it is common practice in flow simulation to save only every n-th computed time step. This means that very often data sets have good resolution in space, but not in time. To make smooth animations of core lines possible, interpolation between two time steps is needed. Higher-order instead of linear interpolation can be used, which on average increases accuracy and reduces visualization artifacts, but comes at higher computational cost and can introduce other artifacts due to overshooting.

With SPH data we have to deal not only with field data, but also with particle positions, the interpolation of which is especially important because it affects not only the particle paths but the reconstruction of all other data channels as well. This is nicely illustrated in Fig. 2 where linear interpolation leads to a water volume that incorrectly fills the hollow roll formed by a wave turning over. A consequence of this incorrectly enlarged domain is that a vortex core line is falsely detected near the center of the roll. When animated, such an "invented" feature pops up only at interpolated, but not at original, time steps.

With cubic interpolation, interpolated particle paths correctly remain inside the water volume and thus the popping artifacts are avoided. For this, it is sufficient to apply cubic interpolation to positions, while field data can be interpolated linearly. The advantage of this scheme is that it can be realized with no more than two time steps simultaneously kept in memory. This is achieved by using cubic Hermite interpolation based on particle positions and on particle velocities which serve as the first derivatives. The resulting particle paths are C1 and more consistent with the particle velocities. This way, a smooth (C^1) trajectory is obtained and moreover, the problem of overshooting is avoided by not applying polynomial interpolation to more than two data points.

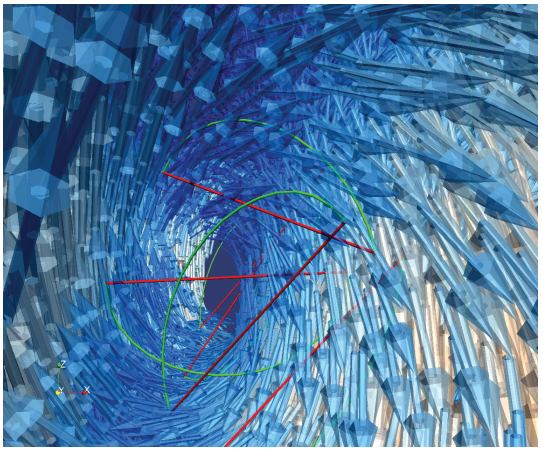

Fig. 2. Importance of cubic interpolation of particle paths. Arrows indicate positions and velocities of all particles and show a region of air under the wave rolling over. Red lines mark linearly interpolated paths of some of the particles, while green curves indicate the corresponding cubically interpolated paths.

4 Unsteady extensions of the Sujudi-Haimes vortex core line criterion

In this section we compare two recent extensions of the Sujudi-Haimes criterion for vortex core lines to unsteady flow. In its original formulation this criterion requires that the velocity gradient has one real and two complex eigenvalues and that the real eigenvector is aligned with the velocity vector. It was discovered later [22] that the second part of the criterion can reformulated as $(\nabla\mathbf{u})\,\mathbf{u} \parallel \mathbf{u}$ where the term on the left can be identified as the *acceleration* of a steady flow. It is obvious that this expression lacking any temporal derivatives cannot correctly describe a vortex axis unless the flow is steady or at least quasi-steady. A straightforward extension is to replace the steady acceleration by the true acceleration $\mathbf{a} = (\nabla\mathbf{u})\,\mathbf{u} + \partial\mathbf{u}/\partial\mathbf{t}$. In the case of a steady flow, this is consistent with the original Sujudi-Haimes criterion, while for unsteady flow it was shown to give better results [4] in the sense that they are more consistent with vortices reported by the λ_2 criterion.

Independently, Weinkauf et al. [27] found a different extension by taking Sujudi and Haimes' initial motivation from 3D to the 4D space-time domain. Instead of identifying the line about which the flow spirals, they identified a "plane of non-swirling flow" in the 4D domain. They came up with criteria for vortex centers in 2D and vortex axes in 3D for both steady and unsteady flow. We discuss these four criteria in the following two subsections, and we give for each of them a second equivalent formulation which we feel is more intuitive and which also reveals some close relations.

4.1 Vortex centers in 2D

2D steady flow. In 2D steady flow a vortex center can be defined as a critical point of the velocity field $\mathbf{u}(\mathbf{x},\mathbf{y})$ of type focus or center. An equivalent formulation is: *The velocity vector is zero and the velocity gradient has two complex eigenvalues.*

2D unsteady flow. Weinkauf's criterion [27] is the Sujudi-Haimes applied to the 3D steady space-time flow (encoding the given unsteady 2D flow):

$$\mathbf{p}(x,y,t) = \begin{pmatrix} \mathbf{u}(x,y,t) \\ 1 \end{pmatrix} = \begin{pmatrix} u(x,y,t) \\ v(x,y,t) \\ 1 \end{pmatrix} \tag{17}$$

We observe that the Jacobian of $\mathbf{p}$

$$\mathbf{J_p} = \begin{pmatrix} \nabla\mathbf{u} & \mathbf{u}_t \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} u_x & u_y & u_t \\ v_x & v_y & v_t \\ 0 & 0 & 0 \end{pmatrix} \tag{18}$$

(with subscripts denoting derivatives) has a zero eigenvalue. The Sujudi-Haimes criterion demands that this is the only real eigenvalue and that $\mathbf{p}$ is the associated eigenvector. Hence we have

$$\mathbf{J_p}\mathbf{p} = 0 \cdot \mathbf{p} \tag{19}$$

which implies

$$(\nabla\mathbf{u})\,\mathbf{u} + \mathbf{u}_t = 0 \tag{20}$$

An equivalent formulation is therefore: *The acceleration vector is zero and the velocity gradient has two complex eigenvalues*.

Consistency with steady case. It is easy to see that the definition for the unsteady case is consistent with the steady case when applied to steady flow. The acceleration vector is $(\nabla\mathbf{u})\,\mathbf{u}$ in the steady case. This is zero if $\mathbf{u}$ is.

4.2 Vortex axes in 3D

3D steady flow. In 3D steady flow a vortex axis can be defined by the Sujudi-Haimes criterion which requires that $\nabla\mathbf{u}(\mathbf{x},\mathbf{y},\mathbf{z})$ has a single real eigenvector and that this is aligned with $\mathbf{u}(\mathbf{x},\mathbf{y},\mathbf{z})$. An equivalent formulation is: *The velocity vector is either zero or along the single real eigenvector of the velocity gradient.*

3D unsteady flow. In 3D, Weinkauf's criterion [27] (equation 28) is that the vector

$$\begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} - f_4\mathbf{u} \tag{21}$$

is the single real eigenvector of the velocity gradient. Here the vector $\mathbf{f}$ denotes the eigenvector of $\mathbf{J_p}$ belonging to the zero eigenvalue:

$$\mathbf{J_p}\mathbf{f} = \begin{pmatrix} \nabla\mathbf{u} & \mathbf{u}_t \\ 0 & 0 \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix} = \mathbf{0} \tag{22}$$

From (22) follows

$$(\nabla\mathbf{u}) \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} + f_4\mathbf{u}_t = 0 \tag{23}$$

and therefore the identity

$$(\nabla\mathbf{u}) \left(\begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} - f_4\mathbf{u} \right) = -f_4\,(\mathbf{u}_t + (\nabla\mathbf{u})\,\mathbf{u}) \tag{24}$$

in which, by (21), the left hand side is an eigenvector of $\nabla\mathbf{u}$. An equivalent formulation is therefore: *The acceleration vector is either zero or along the single real eigenvector of the velocity gradient*.

Consistency with steady case. Again, it is easy to show that the definition for the unsteady case is consistent with the steady case when applied to steady flow. The acceleration vector is $(\nabla\mathbf{u})\,\mathbf{u}$ in the steady case. This is an eigenvector of $\nabla\mathbf{u}$ if $\mathbf{u}$ is.

4.3 Comparison of criteria

The above reformulation of the vortex axis criterion by Weinkauf et al. leads to two interesting observations. Firstly, it shows that this definition of a vortex axis in 3D unsteady flow is Galilean invariant. This is because both the acceleration vector and the velocity gradient are Galilean invariant. Secondly, it reveals a close relation with Fuchs et al.'s unsteady extension of the Sujudi-Haimes criterion [4].

Given a velocity field $\mathbf{u}(\mathbf{x},t)$ and a spatial location $\mathbf{x}$ where the velocity gradient has a single real eigenvalue (and two complex ones), let $\mathbf{e_r}$ denote the corresponding eigenvector, and let $\mathbf{a}$ denote the acceleration vector $(\nabla\mathbf{u})\,\mathbf{u} + \mathbf{u}_t$. With these three vectors the following criteria for vortex core lines can now be expressed:

$\mathbf{u} \parallel \mathbf{e_r}$	Sujudi and Haimes
$\mathbf{u} \parallel \nabla\mathbf{u}\cdot\mathbf{u}$	equivalent to Sujudi-Haimes
$\mathbf{u} \parallel \mathbf{a}$	Fuchs et al.
$\mathbf{a} \parallel \mathbf{e_r}$	Weinkauf et al.

Of all these, Weinkauf et al.'s $\mathbf{a} \parallel \mathbf{e_r}$ is the only predicate which is Galilean invariant, which at least theoretically is an advantage. On the other hand, Fuchs et al.'s $\mathbf{u} \parallel \mathbf{a}$ can be viewed as the more immediate extension to unsteady flow. In practice, the two criteria were found to behave quite similarly and depending on the application one or the other might have slight advantages. We provide a comparison based on unsteady SPH simulations in the results section.

5 Results

5.1 Implementation details

Eigenvector-free formulations. We chose the eigenvector-free formulations of the Sujudi-Haimes and Weinkauf criteria (see Table 1). This way the calculation of the second vector reduces to a matrix-vector product, which is faster than explicitly computing eigenvectors.

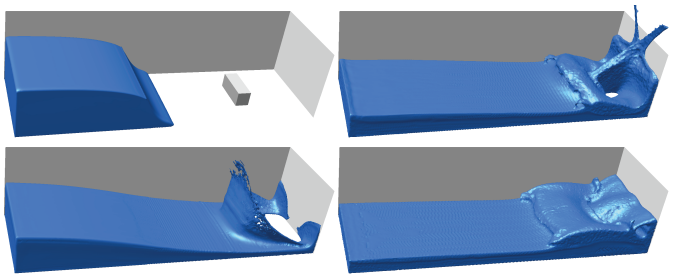

Fig. 3. The Dam breaking simulation data. The free surface is shown at time 0.2, 0.7, 1.2, and 1.7.

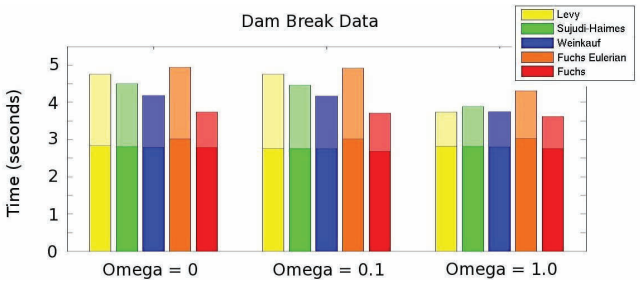

Fig. 4. Total time (seconds) for full feature extraction for the dam break data set. An average over five time steps has been taken, and temporal coherence was disabled. Three levels of candidate filtering based on vortex strength ω are shown. The bars are broken up to show the time taken by the candidate selection process (bottom) and the tracing step (top). It clearly shows that for large data sets and adequate filtering, the process is bound by the amount of time taken finding candidates.

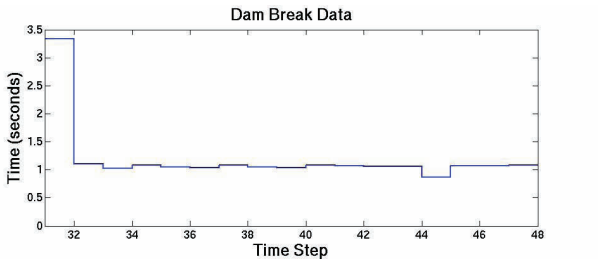

Fig. 5. Impact of exploiting temporal coherence (criterion of Fuchs et al.). The first time step is searched fully, while for subsequent time steps temporal coherence is used. Total CPU seconds for a range of time steps of the dam breaking data.

Eulerian vs. Lagrangian calculation of acceleration. Acceleration data, as is needed for the criteria of Fuchs et al. and Weinkauf et al., is usually not present in simulation data. Naively, it can be generated using the Eulerian formulation $\mathbf{a} = (\nabla \mathbf{u})\mathbf{u} + \frac{\partial \mathbf{u}}{\partial t}$. This approach however requires expensive SPH interpolations in three different time steps, because for computing a finite difference, the velocity has to be evaluated at a given point in space in two additional time steps. A better approach is to calculate accelerations from particle attributes. Since SPH works in the Lagrangian frame, this operation can be done on the raw particle data. For the k-th particle, acceleration is calculated as $\mathbf{a}_k(t) = \frac{\partial}{\partial t}\mathbf{u}_k(t)$ by using a finite difference. This calculation can be done off-line as a form of preprocessing which then gives an additional data channel in the input data, or it can be done on the fly very quickly. This data is treated like any other raw data quantity. The resulting core lines are not only more smooth compared to core lines generated from Eulerian acceleration, they are also significantly faster to generate, as is shown in Fig. 4.

Candidate filtering. As a means to reduce the number of candidates, we explored to filter them and discard those with a vortex strength ω below a given threshold. Vortex strength measures radians per second, but by multiplying with a ratio L/U of a typical length over a typical speed it can be nondimensionalized. For the Dam data $L/U \approx 8.5/3.0$ and for the Tsunami data $L/U \approx 3.2/1.0$, therefore a threshold of 1.0 roughly means a winding angle of π per typical length. We did not observe any loss of features when using this threshold. Fig. 4 and 11 show the savings in computing time for $\omega = 0.0$ (no filtering), 0.1, and 1.0.

Software. The vortex extraction software has been implemented as a module on top of `ParaView` [6]. It can be downloaded from the URL `http://graphics.ethz.ch/research/visualization/sphvis.php`. The temporal interpolation code has been implemented as a separate `ParaView` filter so that it can be reused in conjunction with other filters and is being contributed to the core of the `ParaView` software package. It can be found under `Filters`→ `Temporal`→ `Temporal Interpolator (Particles)`.

All benchmarks were run on an Intel Q6600 quad core processor with 3 GB of RAM running Linux.

5.2 Dam breaking simulation data set

Our first test data set is an SPH simulation of the SPHERIC dam break case [7]. It has 670,575 fluid particles and 87 time steps, and the cubic spline kernel has been used. Solid boundaries are modeled with solid particles, while the air contains no particles.

Qualitative comparison of vortex core line methods. The well known vortex core line methods by Levy et al. and by Sujudi and Haimes are known to often produce quite different results. In this data set, they behave remarkably similar (see Fig. 6). Interestingly, the two unsteady extensions of Sujudi-Haimes produce significantly different results, but among themselves they are quite consistent. This is an indication that the methods designed for steady flow are not appropriate for flow with strongly unsteady characteristics.

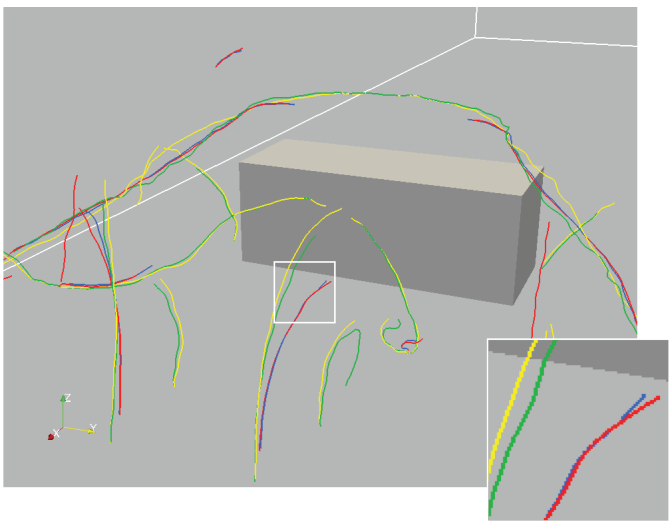

Fig. 6. Comparison of vortex core line criteria, Levy et al. (yellow), Sujudi and Haimes (green), Weinkauf et al. (blue), Fuchs et al. (red). Closeup of the vortex analyzed in detail in the accompanying video. The vortex chosen is one of the quasi-symmetric pair of strongest vortices. Results of the methods designed for steady / unsteady flow are pairwise similar.

Direct vs. resampling-based extraction. As a verification of our predictor-corrector method, we compared results with those obtained from velocity data resampled on a uniform grid. In Figs. 7 and 8 such a comparison is shown for a grid of roughly the same number of nodes (660,275) as there are particles. The results are consistent for all larger and stronger vortices. Feature extraction from resampled data took 1.6 seconds, which is about 50 percent higher than direct

grid nodes	grid spacing	mean error	std. dev.
86,346	0.0264	0.00409	0.00196
660,275	0.0132	0.00305	0.00148
4,351,760	0.0066	0.00217	0.00107

Table 2. Error caused by resampling

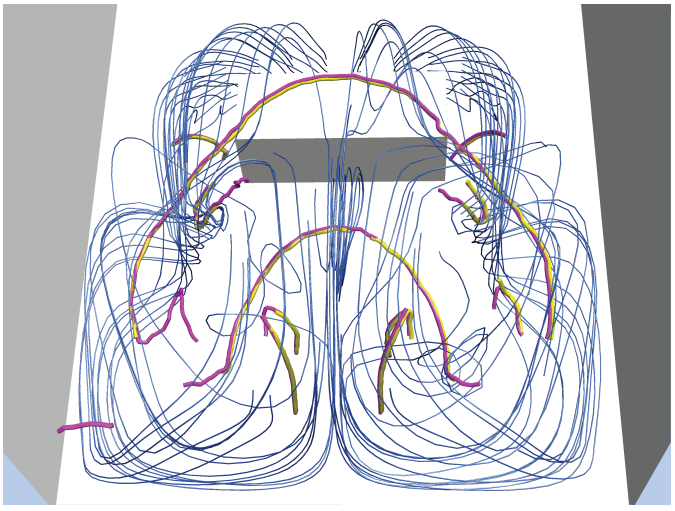

Fig. 7. Vortex core lines directly computed from SPH data (yellow), and from uniformly resampled data (magenta). Levy's criterion (used in both cases) extracts vortex axes in instantaneous velocity field, indicated also by streamlines (dark blue).

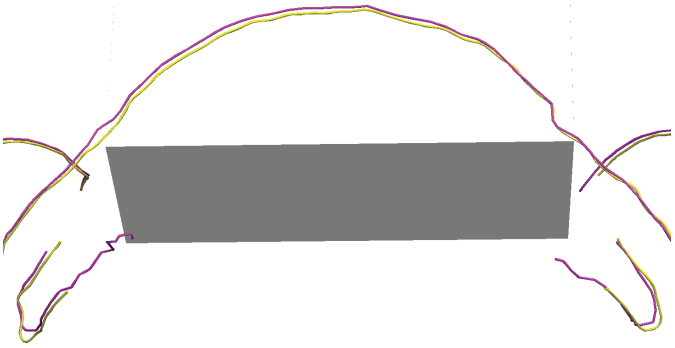

Fig. 8. Close-up of Fig. 7, core lines only. Results show good consistency, with the direct method (yellow) being slightly less noisy.

extraction from SPH data with temporal coherence enabled. In addition, the resampling took 7.2 seconds which, in principle, could also be reduced by exploiting temporal coherence. If temporal coherence is not exploited, our method requires roughly half as much time as the resampling-based method (see Fig. 4). When comparing a resampling-based method with a direct one, a point can be found where the resampling frequency is low enough to make the resampling-based method the faster one [14]. In our case, to reach this point, the grid must be chosen even coarser than we did in the above experiment. That is, there would be fewer sampling points than particles, and consequently the quality of the result would be inferior to that in Fig. 7. For a quantitative assessment of the quality of a feature point, we applied correction steps (Section 3.3) and measured the length of the total correction. As can be seen from Table 5.2 the mean error is roughly a quarter of the grid spacing, and it is therefore very costly to reduce it. This allows us to conclude that resampling is useful only for previewing purposes.

5.3 Tsunami data set

We tested our algorithm also on a second data set, an SPH simulation of the creation of a tsunami [19]. In this simulation, a wedge is sliding downward which is idealizing a section of earth falling. The number of fluid particles is 58,674 (such low numbers are not uncommon in hydrodynamics, in contrast to astrophysics). The number of time steps is 227, and the kernel is a quadratic function within a support radius of $2h$. This data set exhibits a vortex structure above the wedge, moving downward with the wedge while deforming. Again, we observe a clear difference between the features obtained with a steady and an unsteady criterion (see Fig. 9) throughout most of the time steps. The steady criterion (method by Levy et al.) reports a single vortex, however, its middle part is neither in agreement with the unsteady variant nor with the λ_2 isosurface. The unsteady criterion (Fuchs et al.) reports only the two end pieces of that feature, which is consistent with both the steady criterion and the λ_2 isosurface. That the feature does not follow the λ_2 isosurface along the edge of the wedge could be an effect of the low number of particles. But it is also important to notice that λ_2 cannot be taken as a ground truth for vortex detection.

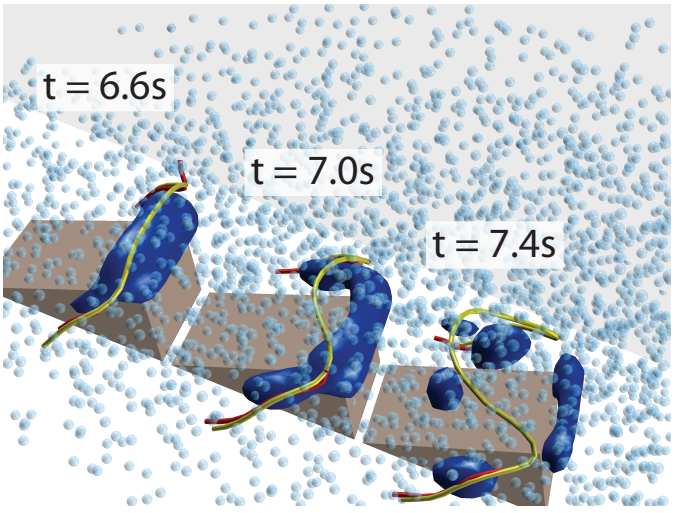

Fig. 9. Three time steps of the tsunami simulation data set. Vortex core lines according to Levy's criterion (yellow) are connected, while those according to Fuchs et al. (red) consist of two shorter pieces. Isosurfaces of λ_2 confirm the existence of a vortex, but not the exact location of either type of core lines. The blue spheres represent (every tenth of) the SPH particles at the first of the three time steps.

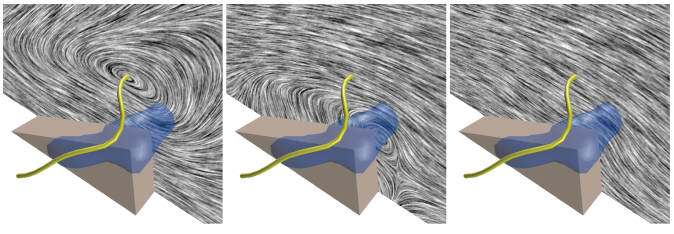

Fig. 10. LIC images of the tsunami velocity field at t=7.0s (the middle time step of Fig. 9) at the symmetry plane of the domain. Velocities are as seen from a static observer (left), an observer moving with the Levy feature curve (middle), and an observer moving with the wedge and thus with the hypothetical vortex indicated by the λ_2 isosurface (right).

Since both the Levy feature curve and the λ_2 isosurface are quasi-symmetric near the symmetry plane of the domain (and of the wedge), we checked both of these features for being consistent with Robinson's vortex definition [18] according to which spiraling flow behavior must be seen by an observer moving with the vortex core (which is within

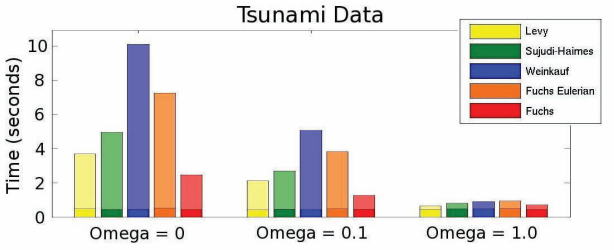

Fig. 11. Total time (seconds) for feature extraction from the Tsunami data set, under the same conditions as for the dam data in Fig. 4. Because there are a lot of small velocity magnitudes, filtering candidates based on ω is essential.

the symmetry plane). The LIC images in Fig. 10 show that spiraling flow exists at the intersection of both the feature curve and the isosurface, but only if the observer moves at a speed different from that of the core of the hypothetical vortex. As a consequence, there is no strong indication of a single connected vortex, and in that sense, the unsteady criterion gives the better results than the steady one.

6 Conclusion

This paper presents a new method for finding vortex core lines in SPH data on the basis of the parallel vectors operator. We have shown how our predictor-corrector approach can make use of the SPH representation to efficiently generate high quality output. The method was then extended to detect loops in vortex core lines and to exploit temporal coherence, speeding up the extraction process significantly. The described approach is straight-forwardly extended to other mesh-free data as long as kernels are radially symmetric and given analytically such that derivatives can be efficiently evaluated. The method is also extendable to other features than vortex core lines. Height ridges, e.g., can be computed by defining the vectors $\mathbf{v}$ and $\mathbf{w}$ of Section 3 appropriately, which involves the Hessian of the given scalar field.

We also contributed a theoretical comparison of two unsteady extensions of the method of Sujudi and Haimes and an analysis of the results obtained with these variants. We found that the two criteria by Weinkauf et al. and by Fuchs et al. produce very similar results. We interpret this result in the way that in practice, Galilean invariance seems less important than the use of a criterion based on the true (unsteady) acceleration vector. Because the criterion by Fuchs et. al also performed better in terms of runtime performance than the other criteria, we can conclude that this criterion should be preferred over the others for visualizing unsteady velocity fields.

What is still a matter of future research is to give a physical interpretation of these two criteria that were derived largely by analogy. We would also like to investigate tolerant filtering not only in space, but also in time, to further reduce flickering of features in animations. One way of achieving this would be to track entire features using corrector steps on the vertices of a feature line of the previous time step. This will require careful handling of merging and splitting of features.

Acknowledgments

The authors would like to thank Jean-Christophe Marongiu for the "dam breaking" data set, Ben Rogers for the tsunami data set, Yun Jang for discussions, and the anonymous reviewers for their helpful comments. The project SemSeg acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number 226042.

References

[1] D. C. Banks and B. A. Singer. A predictor-corrector technique for visualizing unsteady flow. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):151–163, 1995.

[2] J. Biddiscombe, D. Graham, P. Maruzewski, and R. Issa. Visualization and analysis of SPH data. *ERCOFTAC Bulletin, SPH special edition*, 76:9–12, 2008.

[3] Y. Chen, J. Cohen, and J. Krolik. Similarity-guided streamline placement with error evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1448–1455, 2007.

[4] R. Fuchs, R. Peikert, H. Hauser, F. Sadlo, and P. Muigg. Parallel Vectors Criteria for Unsteady Flow Vortices. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):615–626, 2008.

[5] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamic: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, 1977.

[6] A. Henderson. *ParaView Guide, A Parallel Visualization Application*. Kitware Inc. (http://www.paraview.org), 2005.

[7] K. M. T. Kleefsman, G. Fekken, A. E. P. Veldman, B. Iwanowski, and B. Buchner. A volume-of-fluid based simulation method for wave impact problems. *Journal of Computational Physics*, 206(1):363–393, 2005.

[8] R. S. Laramee, D. Weiskopf, J. Schneider, A. Graz, and H. Hauser. Investigating swirl and tumble flow with a comparison of visualization techniques. In *In Proceedings IEEE Visualization 04*, pages 51–58, 2004.

[9] Y. Levy, D. Degani, and A. Seginer. Graphical visualization of vortical flows by means of helicity. *AIAA Journal*, 28:1347–1352, 1990.

[10] L. B. Lucy. A numerical approach to testing the fission hypothesis. *The Astronomical Journal*, 82(12):1013–1924, 1977.

[11] J. C. Marongiu, F. Leboeuf, and E. Parkinson. Numerical simulation of the flow in a Pelton turbine using the meshless method SPH and a new simple solid boundary treatment. *Proc. of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, 221(6):849–856, 2007.

[12] H. Miura and S. Kida. Identification of tubular vortices in turbulence. *Journal of the Physical Society of Japan*, 66:1331–1334, 1997.

[13] J. J. Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68:1703–1759, 2005.

[14] B. Nelson and R. M. Kirby. Ray-Tracing Polymorphic Multidomain Spectral/hp Elements for Isosurface Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):114–125, 2006.

[15] R. Peikert and M. Roth. The "Parallel Vectors" Operator - A Vector Field Visualization Primitive. In *Proc. IEEE Visualization*, pages 263–270, 1999.

[16] R. Peikert and F. Sadlo. Height Ridge Computation and Filtering for Visualization. In I. Fujishiro, H. Li, and K.-L. Ma, editors, *Proceedings of Pacific Vis 2008*, pages 119–126, 2008.

[17] D. J. Price. SPLASH: An interactive visualisation tool for Smoothed Particle Hydrodynamics simulations. *Publications of the Astronomical Society of Australia*, 24:159–173, 2007.

[18] S. Robinson. Coherent Motions in the Turbulent Boundary Layer. *Annual Rev. of Fluid Mechanics*, 23:601–639, 1991.

[19] B. D. Rogers and R. A. Dalrymple. SPH Modeling of Tsunami Waves. In *Advanced Numerical Models for Simulating Tsunami Waves and Runup*, pages 75–100. World Scientific Publishing, 2008.

[20] P. Rosenthal and L. Linsen. Smooth surface extraction from unstructured point-based volume data using pdes. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1531–1546, 2008.

[21] P. Rosenthal, S. Rosswog, and L. Linsen. Direct Surface Extraction from Smoothed Hydrodynamics Simulation Data. In *Fourth High-end Visualization Workshop*, pages 50–61. Lehmanns Media - LOB, 2007.

[22] M. Roth and R. Peikert. A Higher-Order Method for Finding Vortex Core Lines. In *Proceedings of the conference on Visualization '98*, pages 143–150, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[23] D. Sujudi and R. Haimes. Identification of Swirling Flow in 3D Vector Fields. Technical Report 95-1715, AIAA, 1995.

[24] J. Sukharev, X. Zheng, and A. Pang. Tracing parallel vectors. *Visualization and Data Analysis 2006*, 6060(1):682–695, 2006.

[25] H. Theisel, J. Sahner, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Extraction of parallel vector surfaces in 3D time-dependent fields and application to vortex core line tracking. In *Proc. IEEE Visualization 2005*, pages 631–638, October 2005.

[26] A. Van Gelder and A. Pang. Using PVsolve to Analyze and Locate Positions of Parallel Vectors. *IEEE Transactions on Visualization and Computer Graphics*, 15(4):682–695, 2009.

[27] T. Weinkauf, J. Sahner, and H. Theisel. Cores of Swirling Particle Motion in Unsteady Flows. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1759–1766, 2007.