Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2002)
D. Ebert, P. Brunet, I. Navazo (Editors)

Vortex Tracking in Scale-Space

Dirk Bauer and Ronald Peikert

Computer Graphics Lab, Dept. of Computer Science, ETH Zürich, Switzerland
{bauer,peikert}@inf.ethz.ch

Abstract
Scale-space techniques have become popular in computer vision for their capability to access the multi-scale information inherently contained in images. We show that the field of flow visualization can benefit from these techniques, too, yielding more coherent features and sorting out numerical artifacts as well as irrelevant large-scale features. We describe an implementation of scale-space computation using finite elements and show that performance is sufficient for computing a scale-space of time-dependent CFD data. Feature tracking, if available, allows to process the information provided by scale-space not just visually but also algorithmically. We present a technique for extending a class of feature extraction schemes by an additional dimension, resulting in an efficient solution of the tracking problem.

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Applications; I.4.7 [Image Processing and Computer Vision]: Feature Measurement - Feature Representation; J.2 [Applications]: Physical Sciences and Engineering - Engineering

1. Introduction

Automatic extraction of features is a promising strategy to cope with the large amount of data produced by time-dependent CFD (computational fluid dynamics) simulations. Computation time for this type of simulations is typically in the order of days, which justifies the time spent on post-processing the data by extracting features in a batch run. Automatic feature extraction can achieve data reductions of up to 1:10000 [4] while still permitting the visualization of essential parts of the flow. It can be used for visually browsing through the datasets or it can be combined with other types of visualization. However, the implementation of this strategy leads to the following problems:

a. Many flow features are of fractal nature, that means that there is no unique definition of their feature size. Depending on the observation scale, different sets of features can be observed. Usually, the scale is implicitly defined when an extraction method is designed. It would be preferable to let the user specify the scale interactively while viewing the data.

b. Most methods require numerical computation of first- and second-order derivatives, which causes the data to be roughened. Smoothing the data can reduce this effect, but it is not trivial to find an appropriate smoothing kernel when dealing with irregular grids and highly varying cell sizes.

c. When features are extracted from time-dependent data, animating them can cause popping effects with features suddenly appearing or disappearing. This can be alleviated by incorporating temporal in addition to spatial smoothing.

In the field of computer vision, where feature extraction has been practised for a longer time, *scale-space* techniques have been successfully applied. These techniques smooth the data using Gaussian kernels, the standard deviation σ of which can be any positive real number. This *scale axis* plus the spatial axes span the scale-space. Features thus not only have spatial extents but also a certain scale extent. They can therefore be searched and found in scale-space. This technique has the potential to solve the problems mentioned above. However, for flow visualization applications, some additional adaptations must be made: Firstly, a discretized smoothing operation must be provided also for curvilinear and unstructured grids. And secondly, special attention has to be paid to boundary treatment, due to the relatively low resolutions in CFD datasets (often having interior boundaries) and the importance of the flow behavior near material boundaries.

Many of the published multi-scale techniques are multi-resolution as well, e.g. [9] and [19] performed feature extraction from volumetric data in scale-spaces based on wavelets. However, to keep the spatial resolution, we do not construct a multi-resolution pyramid. Also, such a pyramid would yield a coarse sampling along the scale axis. Yet for the purpose of tracking, it is preferable to have no prescribed sampling.

The idea of using the scale-space for visualization of vector fields is not new. Diewald et al. [1] demonstrate the usefulness of anisotropic diffusion for the visualization of vector fields. By successively smoothing the data the scale-space can be visually explored. Our goals are, beyond visual exploration, to extract features as geometric objects and to improve this process by exploiting the multi-scale nature of the features.

Scale-space analysis can enhance flow visualization in many ways:

a. At larger scales, the set of features is reduced to fewer and clearer features. This is particularly noticeable for features defined by second derivatives [10,13]. The need for feature simplification has been recognized before, leading to simplification strategies for special purposes proposed by several authors [1,17,21].

b. It becomes possible to focus on features of a certain scale.

c. Tracking features along the scale-axis allows to visualize them with the positional accuracy of small scales and, at the same time, to derive connectivity information from larger scales.

d. Selective visualization can be done by picking an individual feature at a larger scale which is then tracked to a smaller scale and finally tracked through time.

All said above applies to flow features of any dimensionality. However, we will focus on 1D (line-type) features, for the reason that the CFD datasets computed by our industry partners are mostly visualized for the purpose of studying vortices. Vortices reduce a machine's efficiency by binding energy. Also, an unstable vortex can interact with machine parts, producing undesired effects like material abrasion or resonance.

The following section provides the definition of the linear scale-space. Its numerical computation in typical CFD grids is discussed in Section 3. Section 4 recapitulates a general vortex extraction algorithm previously introduced [14] and describes adaptations and simplifications to be made in the context of a scale-space analysis. It is then shown in Section 5 how a 4D extension of this algorithm is able to track vortices in either the temporal or the scale domain. Finally, both numerical and visual results are given.

2. The scale-space

When doing physical measurements or observations of real-world objects, it is impossible to get a perfect representation because the "aperture" of the observation instrument (e.g. a human's eye or a photo camera) cannot be varied arbitrarily, thus the image resolution is limited to a certain range of scales. Such multiscale representations of images have widely been used since the beginning of the 1970s, but it is no trivial task to track image structures across different levels of scale, nor to distinguish significant image features from noise. A major breakthrough was the introduction of the scale-space theory by Witkin [20] and Koenderink [5] in 1983/84. They define the inner scale as the smallest level-of-detail we can resolve and represent in our data (e.g. one cone/rod of our retina, or one image pixel), the outer scale as the largest possible structure (our field of view, or the whole image).

Florack et al. [3] proved that convolution of a data signal $u(\boldsymbol{x})$ with the (n-dimensional) Gaussian kernel

$$G(\boldsymbol{x}, \sigma) = (2\pi\sigma^2)^{-n/2} e^{-\boldsymbol{x}^2/(2\sigma^2)} \qquad (1)$$

is the unique family of "aperture" functions to meet the following requirements:

- *Linearity:* Applying the smoothing function can be interchanged with adding two datasets or multiplying a data set with a scalar.
- *Shift invariance:* There is no preferred location.
- *Scale invariance:* There is no preferred size.
- *Isotropy:* There is no preferred direction.
- *Semigroup property:* The set of smoothing functions forms a commutative semigroup w.r.t composition. A semigroup is closed under the operation, is associative and has a zero element.

According to Lindeberg [7], the *linear scale-space* of a given physical space $X \subseteq \boldsymbol{R}^n$ with a data field $u(\boldsymbol{x})$ is defined to be the space $X \times \boldsymbol{R}_0^+$ with data

$$u(\boldsymbol{x}, s) = u(\boldsymbol{x}) \otimes G(\boldsymbol{x}, \sqrt{s})\,. \qquad (2)$$

Nonlinear scale-spaces can be obtained by relaxing some of these requirements. A famous example is Perona and Malik's [11] scale-space based on anisotropic diffusion. However, we will in the following restrict ourselves to the linear scale-space.

3. Computing the scale-space

Convolving a scalar field *u(x)* with a Gaussian of standard deviation σ is equivalent [5] to solving the heat equation

$$\dot{u}(\boldsymbol{x}, s) = \frac{1}{2}\nabla\bullet\nabla u(\boldsymbol{x}, s) \qquad (3)$$

for time $s = \sigma^2$ and initial condition $u(\boldsymbol{x}, 0) = u(\boldsymbol{x})$.

The reason for calling the time variable *s* rather than *t* is that we want to do scale-space analysis also for time-dependent data such as unsteady flow fields. We then have two orthogonal time axes, namely the "physical time" *t* and the "diffusion time" *s*.

By definition of the scale-space (Eq. (2)), the diffusion time *s* is equal to the scale parameter *s*.

3.1. Regular grids

On a regular grid there are at least three fundamentally distinct algorithms for smoothing data with a Gaussian.

a. The obvious one is to actually compute the convolution with a sampled Gaussian. This can be performed either in the spatial domain or in the frequency domain.

b. The second method is to repeatedly apply for each dimension a filter with weights (1/4, 1/2, 1/4). The weights of the product filter are up to normalization the even rows of the Pascal triangle and thus converge to a Gaussian. This seems to be better suited to the computation of a scale-space where a whole sequence of Gaussians is needed. However, the main problem with this approach is that the number of iterations is proportional to *s* and thus to σ^2.

c. The third method is to solve the heat equation. While this amounts to numerically solving a partial differential equation, it has the advantage to work well also for large σ and also to extend properly to irregular (curvilinear or unstructured) grids.

Only the third method will be further pursued since CFD grids are typically irregular.

3.2. Computing derivatives

Probably all feature extraction techniques, be it in computer vision or in scientific visualization, require in addition to the field data the first and sometimes the second spatial derivatives. The need for smoothing increases with the order of the derivative as it must compensate for the roughening effect of numerical differentiation.

A basic property of the convolution operator is that it commutes with differentiation

$$\frac{\partial}{\partial x}(u \otimes G) = \frac{\partial u}{\partial x} \otimes G = u \otimes \frac{\partial G}{\partial x} \qquad (4)$$

which provides us with three different ways to compute derivatives of the smoothed data. The left term corresponds to differentiating the smoothed data. This is our preferred approach because it is numerically clearly better than smoothing the differentiated data as reflected by the middle term. It also makes differentiation a simpler task which can be done with minimal stencils. The right term corresponds to using sampled derivatives of the Gaussian which are then convolved with the data. This computing strategy makes particular sense in an environment where smoothing is generally done with sampled Gaussians. However, this is not necessarily the best strategy in the context of computing a scale-space, especially if the grid is unstructured.

3.3. Unstructured grids

The grids typically used in turbomachinery CFD consist of hexahedral cells and are either block-structured or unstructured. We will focus on the latter as they comprise the former. The grid cells can directly be used for a finite element (FE) discretization of the heat equation. This type of general hexahedral elements is usually referred to as isoparametric elements [21]. They are unit cubes when expressed in local coordinates ξ, η and ζ.

A natural choice for boundary conditions are symmetry boundary conditions, where the normal derivative is forced to be zero. This is the simplest form of a Neumann boundary condition. It says that diffusion can happen unrestrictedly along the boundary, but no diffusion is allowed across the boundary. The choice of the symmetry boundary condition has a positive effect on the computation: No boundary integrals have to be computed, and as a consequence both matrices in the spatially discretized equation $\boldsymbol{A}\dot{\boldsymbol{u}} - \boldsymbol{B}\boldsymbol{u} = 0$ are symmetric. The advantage of symmetric matrices is, besides reduced storage requirements, the availability of more efficient solvers.

The entries of the element mass matrix [21] are then

$$a_{ij} = \int_0^1\int_0^1\int_0^1 \varphi_i\varphi_j \det\boldsymbol{J}\; d\zeta d\eta d\varsigma \qquad (5)$$

where $\varphi_i(\zeta, \eta, \varsigma)$ and $\varphi_j(\zeta, \eta, \varsigma)$ are the basis functions and

$$\boldsymbol{J} = \frac{\partial(x, y, z)}{\partial(\zeta, \eta, \varsigma)}. \qquad (6)$$

The element stiffness matrix [21] has entries

$$b_{ij} = \int_0^1\int_0^1\int_0^1 (\boldsymbol{J}^{-1}\nabla'\varphi_i) \bullet (\boldsymbol{J}^{-1}\nabla'\varphi_j)\, \det\boldsymbol{J}\; d\zeta d\eta d\varsigma \quad (7)$$

where ∇' is the gradient w.r.t. local coordinates. The integrals Eq. (5) and Eq. (7) are usually computed numerically with a method such as Gauss quadrature.

4. Application of the scale-space to flow visualization

Scale-space analysis can be done in conjunction with any visualization technique that requires some sort of smoothing, in particular with techniques that include derivatives. But like in the field of image processing (e.g. [7]), it is most successfully combined with feature extraction techniques. Features of interest can be of various types and dimensions, but since our main application is extraction of vortex cores from CFD datasets, we will focus on line-type features. The "parallel vectors" operator [14] allows us to specify a variety of line-type features, including vortex core definitions given by Levy et al. [6], by Sujudi and Haimes [16], and by Miura and Kida [10]. The "parallel vectors" algorithm basically computes the set of points where two given vector fields $\boldsymbol{v}$ and $\boldsymbol{w}$ are (anti-)parallel, i.e. there exists some scalar value λ such that $\boldsymbol{v} = \lambda\boldsymbol{w}$. We allow λ to be $\pm\infty$ to include the case where $\boldsymbol{w}$ vanishes. The algorithm can be implemented by scanning all grid faces for intersection points which are then connected to lines. By subsetting these lines the final feature lines are obtained. Subsetting is based on the velocity field $\boldsymbol{u}$. In an abstract sense, the "parallel vectors" method operates on the three fields $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{w}$. But in practice, often one of $\boldsymbol{v}$ and $\boldsymbol{w}$ is the velocity field $\boldsymbol{u}$ itself. Depending on the underlying feature definition, $\boldsymbol{v}$ and $\boldsymbol{w}$ can be various derived fields such as pressure gradient, vorticity or acceleration.

In the context of a scale-space analysis, this procedure can be simplified in two ways:

a. Since data are presmoothed, estimating derivatives (for setting up the derived fields) can be done with a simple scheme and no extra filtering.

b. Subsetting of the lines can be done more automatically and with fewer "quality" parameters. In the case of vortex cores, one parameter turned out to be sufficient, namely the ratio of twist and forward motion. Other parameters used in the original procedure, like deviation of the core tangent from the velocity direction, are no longer needed, since "insignificant" features are eliminated automatically when increasing the scale *s*.

The vortex core extraction algorithm is recapitulated here (see Fig. 1) mainly because it will serve as a basis for our vortex tracking algorithm. Intersection points of core lines with cell faces are computed with the eigenvector method [14] but we changed the procedure for connecting the points. Each intersection point is now stored in an attributed vertex list. Attributes are the value of λ and the quality parameters mentioned above. When all six faces of a cell have been treated, and if the intersection problem did not degenerate, the result is an even number of vertices in the cell, usually zero or two. These vertices are connected, where the ambiguous situation of more than two vertices is resolved heuristically by pairing vertices with similar λ values. Orientation of the line segments is chosen consistently with the mean velocity at their end points. Matching segments in adjacent cells are then connected to polylines as long as the orientation is conserved. Fig. 1 summarizes this procedure in pseudo-code.

```
Initialize a segment list S.
for each grid cell
  Initialize a vertex list L.
  // Find points within the grid cell.
  for each of the 6 cell faces
    Find all points on the face where v and w
      are parallel.
    Insert the points into L, sorted by
      ascending λ.
  next face
  // Connect the points within this cell.
  while |L| >= 2
    Extract the first two points from L and
      connect them to a line segment.
    Orient the line segment consistently with
      the mean velocity at its two end points.
    Add the line segment to S.
  end while
next grid cell
Connect adjacent line segments in S to polylines.
```

Figure 1: *The vortex extraction algorithm.*

When viewing the resulting polylines, they should be filtered based on vertex attributes. Simple thresholding is possible as well as more sophisticated techniques such as hysteresis thresholding. Good results can also be obtained by allowing a certain number of exceptions between two "valid" vertices. Finally, a minimal feature size can be imposed.

5. Feature tracking

For a couple of reasons we need a way to track features from one dataset to another. The obvious application is to track features along the time axis in time-dependent data. But it also makes sense to track features along the scale axis, this way exploring the information contained in the scale-space. An example of this is to interactively select an individual feature at a larger scale and then display it at a smaller scale, where positions are more accurate but where the feature may be broken up into disjoint fragments. In both cases we have the dimensions of the data domain plus an extra dimension which we might call the tracking dimension. The two types of tracking can also be combined, as in the example shown in Fig. 2.

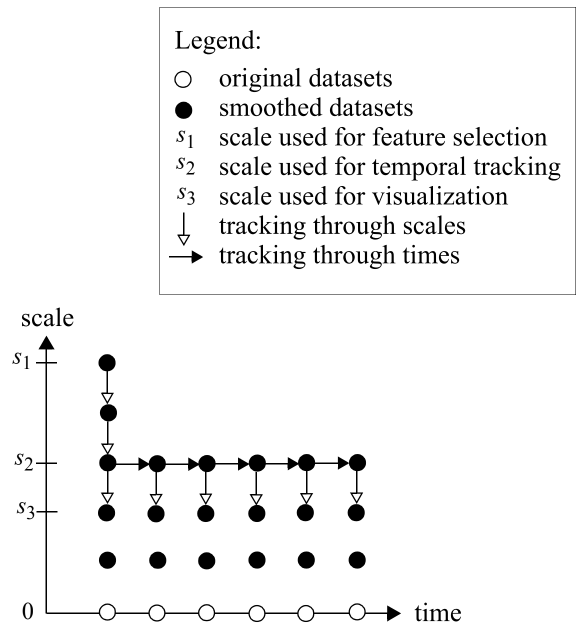

Figure 2: *Example application of tracking in scale and time.*

Most existing tracking methods [15,12] operate on features extracted from single datasets. This approach can be compared to reconstructing an isosurface from given contour plots in a pile of slices. Before the Marching Cubes algorithm [8] was known, isosurface extraction had been approached as a tracking problem with z being the tracking dimension.

The improvement brought by Marching Cubes was to "lift" the contour extraction from 2D to 3D. Instead of working with 2D grid cells, the grid is extended by the tracking dimension and the contour extraction is done for 3D cells. Such a lifting technique can basically be applied whenever a feature extraction is operating on a cell-by-cell basis. An example is the tracking of critical points by Tricoche et al. [18]. In their paper the tracking dimension is time, but it can be treated just like the z dimension in the Marching Cubes example. For simpler notation, we will, in the following, use "time" for the tracking dimension.

Features extracted from a lifted grid get an additional dimension, too. For example, 0D features such as critical points become lines. Tracking is now simply achieved by taking slices of constant time. In an implementation, it is of course more appropriate to represent time as a vertex attribute rather than as an additional coordinate. Then, taking a slice of constant time t means to extract a level set for time t. An advantage of this tracking method is that it needs no heuristics like spatial overlap [15] or shape attributes [12], and still can handle features moving by more than a grid cell per dataset (feature B in Fig. 3). Tracking is correct w.r.t. to linear interpolation of time.

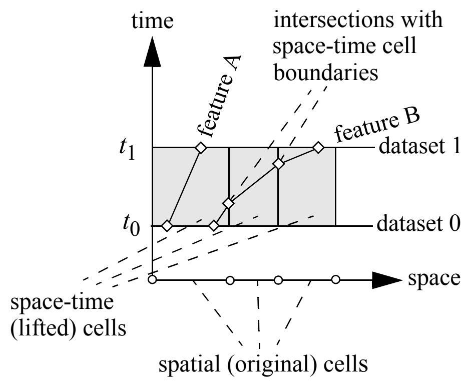

Figure 3: *Lifting of cells and features.*

Since we focus on vortex cores (or more generally on line-type features) in hexahedral grids, lifting generates cells which topologically are 4D hypercubes. A 4D hypercube, or tesseract, has 16 vertices, 32 edges, 24 faces and 8 boundary cubes. Features are lifted, too, namely from 1D to 2D manifolds. It is clear that degeneracies can cause feature dimensions to be other than expected. Implementations have to take this into account. However, this is not a problem of the lifting scheme since the problem is already present in the underlying extraction method.

The algorithmic problem can now be stated as follows: Given two vector fields $\boldsymbol{v}(x, y, z, t)$ and $\boldsymbol{w}(x, y, z, t)$ on the 16 vertices of a hypercube, find the set of points where the two fields are parallel, and output it as a triangle mesh in 3-space. Time should be stored as a vertex attribute, in addition to the attributes of Section 4.

The 16 corners of a hypercube consist of the eight corners of a grid cell at two consecutive times t_0 and t_1. The eight boundary cubes are as follows: two of them are purely spatial (one at time t_0 and one at time t_1), the other six are space-time cubes, each consisting of one cell face at both times (see Fig. 4).

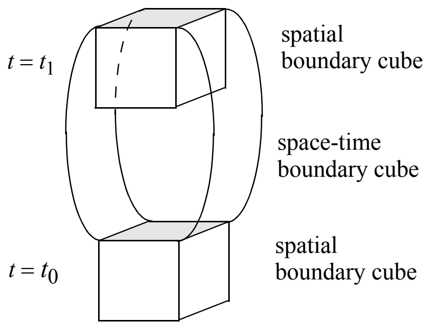

Figure 4: *Three of the eight boundary cubes of a hypercube cell.*

Since all eight boundary cubes have the same data format as a standard grid cell, we can apply to each of them the procedure we used for the cells in Section 4. This leads to the algorithm shown in Fig. 5. The resulting triangle mesh needs again to be post-processed. The minimum is to apply thresholds for vertex attributes. Optional steps can be to compute connected components and/or to extract level sets of time.

```
Read first dataset (time $t_0$).
for i = 1 to #datasets - 1
  Read next dataset (time $t_i$).
  for each grid cell
    Get the data values at all 16 corners of its
      hypercube.
    // = data values at the cell corners
    // for times $t_{i-1}$ and $t_i$.
    for each of the 8 boundary cubes
      // Find line segments in the boundary cube.
      Apply the procedure from Fig. 1.
      Project line segments to 3-space
        and store time as attribute.
    next boundary cube
    // The line segments form closed polygons.
    Triangulate the polygons and add triangles
      to triangle list.
  next grid cell
  Release older dataset (time $t_{i-1}$).
next i
```

Figure 5: *The vortex tracking algorithm.*

6. Results

In this section we provide numerical and visual results based on two time-dependent datasets. The first one is an unsteady simulation (by VA Tech Hydro) of the draft tube of a Francis turbine. The main purpose of the simulation was to predict the so-called vortex rope, a helical vortex having a precession rate of roughly one third of the runner frequency. The computation was done for the original and for an optimized runner geometry. The optimization improved the flow behavior, but its less articulate vortex rope is more difficult to extract. Fig. 7 shows the vortex rope in the original design. The vortex core was extracted based on the definitions by Miura and Kida [10] and by Levy et al. [6]. Because the former works with second derivatives, some smoothing was necessary. The latter produced acceptable results even without smoothing. Regarding the redesigned draft tube (Fig. 8), it is remarkable how the topology of the vortex cores depends on the scale. At the smallest scales, noise is dominating, which disappears at medium scales. At larger scales, a vortex core close to the machine axis appears, describing the global swirl in the draft tube. Tracking the features through different scales allows us to recognize the fragmented cores without sacrificing the positional accuracy (Fig. 9, left).

The second dataset is an unsteady simulation (by Sulzer Pumpen) of a mixed-flow pump operated at 35% of its best efficiency point. Computation on an SGI Power Challenge for 1800 time steps (1-degree increments) took 73.5 CPU-hours. Both datasets are discretized on unstructured hexahedral grids. Temporal tracking of vortex cores generates a description of the vortex motion. In Fig. 9 (right) it is apparent that most of the vortices are quite stable, whereas one large vortex in each diffusor channel oscillates.

In Table 1 we provide performance results for the Gaussian smoothing. In all three grids, the hexahedral cells were used as finite elements discretization with linear isoparametric elements. We list the CPU times for computing the two FE matrices and for smoothing the data in a single step, for three

dataset	nodes	extent	σ	matrix setup	smoothing	
				CPU[s]	CPU[s]	iterations
draft tube	654770	1.33 × 0.59 × 0.72	0.002	531.8	41.2	9
			0.008		88.5	23
			0.032		219.6	62
pump stator	310757	0.50 × 0.50 × 1.54	0.002	242.3	19.2	9
			0.008		31.8	17
			0.032		67.9	40
pump rotor	235223	0.33 × 0.33 × 0.19	0.002	185.4	18.1	12
			0.008		31.2	23
			0.032		61.1	48

Table 1: *Performance analysis of scale-space computation*

values of σ. Computations have been done on an SGI Octane with a single 250 MHz R10000 processor and 640 MB memory.

The FE matrix elements were computed with Gauss quadrature (3x3x3 Gauss points). The linear systems then were solved using the `f11jef` function from the NAG Fortran library. The method used was symmetric Lanczos and preconditioning was done with symmetric successive overrelaxation. Setting up the matrices turned out to be quite expensive. However, the matrices can be used for all datasets related to the same grid, since they contain purely geometric information. Computing times for smoothing seem to scale less than linearly with σ. If an explicit scheme were used, they would scale quadratically because of $s = \sigma^2$ and the limited step size. This certainly demonstrates the value of implicit methods, although for good accuracy the step size must be bounded, too.

Fig. 6 demonstrates the feature simplification aspect of the scale-space approach. Features have been extracted from all three datasets at fixed times but different scales. The number of features is plotted against the standard deviation σ of the

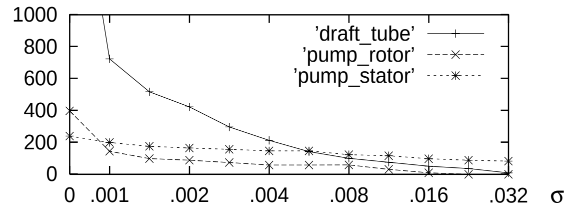

Figure 6: *Number of features at different scales*

Gaussian smoothing kernel. As can clearly be seen, the number of features decreases significantly at higher smoothing levels.

7. Conclusion

We have implemented a method for computing the linear scale-space of unstructured grid data. By controlling a single parameter, the scale, we were able to improve the extraction of features, in particular of features defined in terms of second spatial derivatives. We have also presented a novel 4D tracking method for line-type features which can be used to track vortex cores through different scales carrying over topological information such as connectivity. The same algorithm can be used for temporal tracking as is needed for selective visualization of features in time-dependent data. Our implicit tracking method can reliably treat fast-moving features and is therefore a good alternative to proximity-based methods.

Our application-specific flow fields have quasi-periodically moving features. It is therefore not necessary to explicitly deal with bifurcations. Future work should address this issue, giving more flexibility when visualizing general types of flow fields. The basic information on bifurcations is provided by our tracking algorithm.

Acknowledgments

This work has been supported by the Swiss Commission for Technology and Innovation, by VA Tech Hydro, Zürich, and by Sulzer Markets and Technology, Winterthur. We thank Peter Arbenz, Armin Friedli, Markus Gross, Tobias Preusser and Martin Rumpf for discussions and Filip Sadlo for programming the FE module.

References

1. W. de Leeuw and R. van Liere. Visualization of Global Flow Structures Using Multiple Levels of Topology. In *Data Visualization '99 (Proc. VisSym '99),* pp. 45-52. Springer Verlag, 1999.
2. U. Diewald, T. Preusser and M. Rumpf. Anisotropic Diffusion in Vector Field Visualization on Euclidean Domains and Surfaces. In *IEEE Transactions on Visualization and Computer Graphics,* pp. 139-149, Apr.-Jun. 2000.
3. L.M.J. Florack, B.M. ter Haar Romeny, J.J. Koenderink and M.A. Viergever. Scale and the differential structure of images. *Image and Vision Computing,* vol. 10, pp. 376-388, 1992.
4. D. Kenwright. Automatic Detection of Open and Closed Separation and Attachment Lines. In *Proceedings of Visualization '98,* pp. 151-158, 1998.
5. J. J. Koenderink. The structure of images. *Biological Cybernetics*., vol. 50, pp. 363--370, 1984.
6. Y. Levy, D. Degani and A. Seginer. Graphical Visualization of Vortical Flows by Means of Helicity. *AIAA* 28(8), pp. 1347-1352, August 1990.
7. T. Lindeberg. *Scale-Space Theory in Computer Vision.* Kluwer Academic Publishers, Boston, 1994.
8. W. Lorensen and H. Cline. Marching Cubes: A High Resolution 3D Surface ConstructionAlgorithm. *Computer Graphic*s, 21(4):163–169, 1987.
9. C. Lürig, R. Grosso and T. Ertl. Combining Wavelet Transform and Graph Theory. In *Visualization in Scientific Computing 1997*, pp. 137-144, Springer Verlag, 1997.
10. H. Miura and S. Kida. Identification of Tubular Vortices in Turbulence. *Journal of the Physical Society of Japan,* vol. 66, nr. 5, pp. 1331-1334, May 1997.
11. P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. In *IEEE Computer Society Workshop on Computer Vision,* (Miami, FL), pp. 16--22, 1987.
12. F. Reinders, F.H. Post and H.J.W. Spoelder. Attribute-Based Feature Tracking. In *Data Visualization '99 (Proc. VisSym '99),* pp. 63-72. Springer Verlag, 1999.
13. M. Roth and R. Peikert. A Higher-Order Method for Finding Vortex Core Lines. In *Proceedings of Visualization '98,* pp. 143-150, Oct. 1998.
14. M. Roth and R. Peikert. The "Parallel Vectors" Operator - A Vector Field Visualization Primitive. In *Proceedings of Visualization '99,* pp. 261-268, Oct. 1999.
15. D. Silver and X. Wang. Volume Tracking. In *Proceedings of Visualization '96,* pp. 157-164, 1996.
16. D. Sujudi and R. Haimes. Identification of Swirling Flow in 3D Vector Fields. *Tech. Report, Dept. of Aeronautics and Astronautics,* MIT, Cambridge, MA, 1995.
17. X. Tricoche, G. Scheuermann, H. Hagen and S. Clauss. Vector and Tensor Field Topology Simplification on Irregular Grids. In *Data Visualization 2001 (Proc. VisSym '01)*, pp. 107-116, Springer Verlag, 2001.
18. X. Tricoche, G. Scheuermann and H. Hagen. Topology-Based Visualization of Time-Dependent 2D Vector Fields. In *Data Visualization 2001 (Proc. VisSym '01)*, pp. 117-126, Springer Verlag, 2001.
19. R. Westermann and T. Ertl. A Multiscale Approach to Integrated Volume Segmentation and Rendering. *Computer Graphics Forum,* vol. 16, nr. 3, pp. 117-127, September 1997.
20. A.P. Witkin. Scale-space filtering. In *Proc. International Joint Conference on Artificial Intelligence*, (Karlsruhe, Germany), pp. 1019-1023, 1983.
21. P.C. Wong, H. Foote, R. Leung, E. Jurrus, D. Adams and J. Thomas. Vector Fields Simplification - A Case Study of Visualizing Climate Modeling and Simulation Data Sets. In *Proceedings of Visualization '00,* pp. 485-488, Oct. 2000.
22. O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method, vol. I.* (Fourth edition). McGraw Hill, 1988.

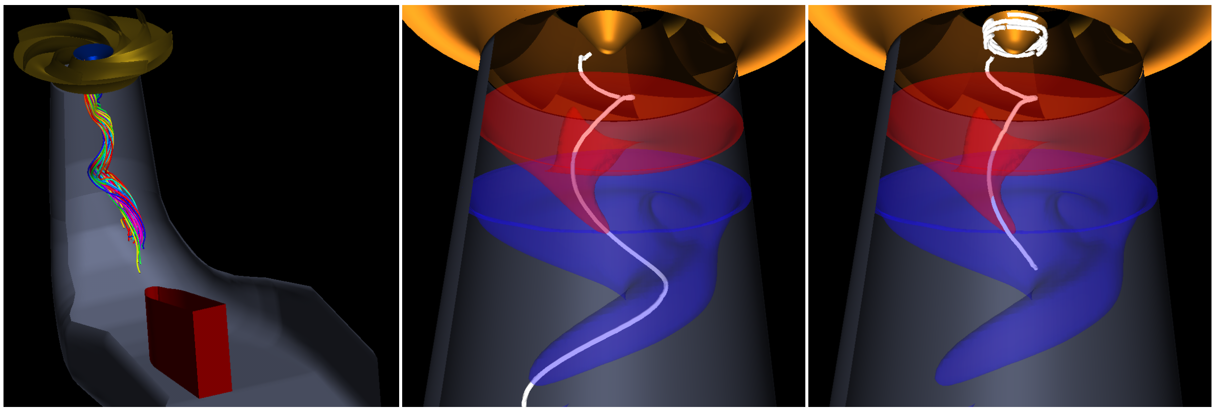

Figure 7: *Original draft tube dataset: geometry and instantaneous streamlines (left). Pressure isosurfaces and pressure valley line indicate the so-called vortex rope. Pressure data have been Gaussian-smoothed with σ=0.008 (middle). Vortex core extraction based on normalized helicity can be successfully applied to unsmoothed data (right).*

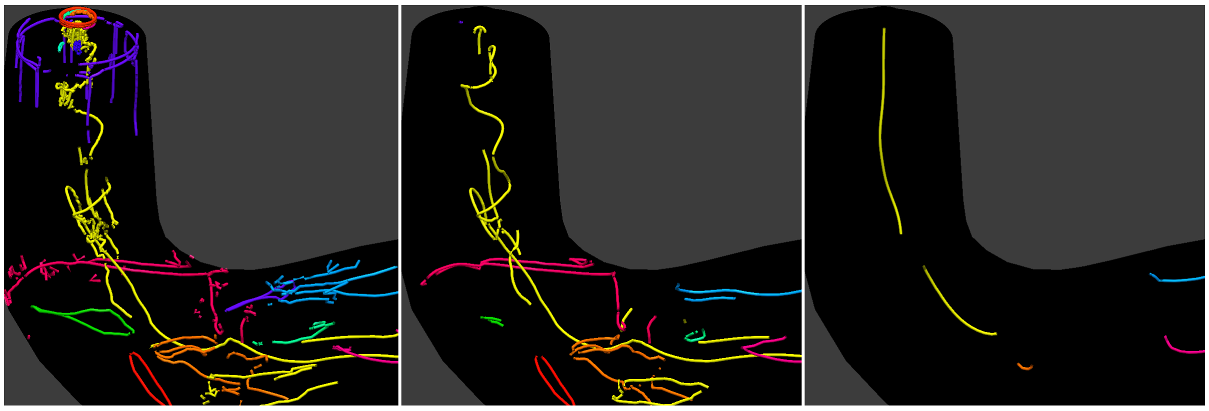

Figure 8: *In the modernized draft tube, vortices are less articulate and harder to extract. Vortex core extraction based on normalized helicity has been applied to unsmoothed velocity field (left) and to Gaussian-smoothed data with σ=0.008 (middle) and σ=0.032 (right). Colors represent connected components of the surface swept by the core lines, see Fig. 9.*

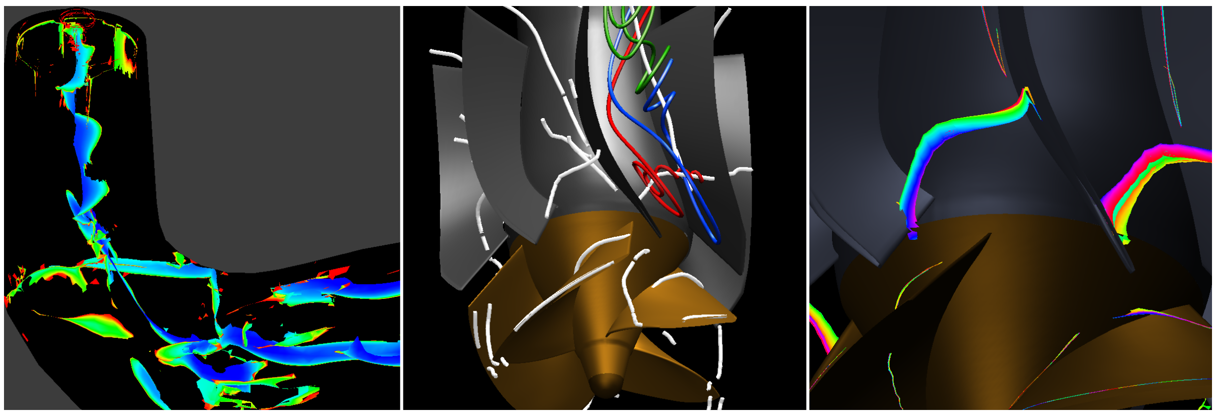

Figure 9: *Vortex cores of Fig. 8 tracked through scales $\sigma \in [0, 0.032]$ (left). Diagonal pump dataset with vortices extracted in runner and diffusor along with a few manually seeded streamlines (middle). Vortex cores tracked temporally for a 90° rotation of the four-bladed runner (right).*