

# Priority Streamlines: A context-based Visualization of Flow Fields

Michael Schlemmer<sup>1</sup>, Ingrid Hotz<sup>2</sup>, Bernd Hamann<sup>3</sup>, Florian Morr<sup>1</sup> and Hans Hagen<sup>1</sup>

<sup>1</sup>Computergraphics and Visualization, DFG International Research Training Group 1131, University of Kaiserslautern, Germany

<sup>2</sup>Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), Germany

<sup>3</sup>Institute for Data Analysis and Visualization and Department of Computer Science (IDAV), University of California, Davis, CA

---

## Abstract

Flow vector fields contain a wealth of information that needs to be visualized. As an extension of the well-known streamline technique, we have developed a context-based method for visualizing steady flow vector fields in two and three dimensions. We call our method "Priority Streamlines". In our approach, the density of the streamlines is controlled by a scalar function that can be user-defined, or be given by additional information (e.g., temperature, pressure, vorticity, velocity) considering the underlying flow vector field. In regions, which are interesting the streamlines are drawn with increased density, while less interesting regions are drawn sparsely. Since streamlines in the most important regions are drawn first, we can use thresholding to obtain a streamline representation highlighting essential features. Color-mapping and transparency can be used for visualizing other information hidden in the flow vector field.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Line and Curve Generation

---

## 1. Introduction

Streamlines are commonly used for visualizing flow vector data. Prior to computer-driven visualization engineers used dyes in water or smoke in air for visualizing flows. Streamlines can be mathematically described, so today they are mostly numerically computed and visualized on computers. They provide a very intuitive representation of flow fields. But especially in the case of 3D fields, they are afflicted with the problem of generating visual clutter and occlusion. To make streamline visualizations also useful for 3D flows it is important to reduce the number of streamline. The information content of the resulting representation strongly depends on the choice of the displayed streamlines. We developed a method to draw context-sensitive streamlines for 2D and 3D flow vector data. With *context-sensitive* we mean that a user can define regions or features of interest, which are highlighted by higher streamline density, filtering out unimportant information. The use of density to highlight important features leaves color and transparency for displaying other information on the streamlines. It also supports a side by side comparison various scalar fields

without occlusion. As we enable the user to define certain priorities for regions or patterns, we call our lines "*priority streamlines*."

Our method is an image based approach, reaching the target density without computing explicit distances of streamlines. Instead the streamline drawing is guided by a comparison of a blurred version of the current image with the target density. This idea is similar to the work of Salisbury et al. [SWHS97] who use orientable textures to render line drawing images. We interpret the tree basic issues for streamline representations, formulated by Verma et al. [VKP00] in the following way:

- *Coverage*: No important flow feature should be missed. Flow features can be topological features as described by Verma et al. [VKP00] but also any other patterns or regions of (high) interest to the user, e.g. of geometric nature or defined by related scalar fields.
- *Uniformity*: This principle argues that streamlines should be distributed more or less uniformly in the final image for better visual interpretation. This is of course a desir-

able criterion, especially for two dimensions. But since for 3D data each viewing direction results in a different density distribution in the final 2D projection we consider the criterion of uniformity as less important to reach our goal.

- *Continuity*: Another goal is to achieve an impression of continuity. This can be done by drawing long streamlines.

As streamline seeding has a major influence on the resulting image many methods optimize streamline seeding. First, Turk and Banks [TB96] focused on the uniformity of streamlines for 2D data, by presenting an image-guided streamline placement method. They reach high quality results, but the method is computationally very expensive. Mao et al. [MHHI98] transferred this method to curvilinear grids. Jobard and Lefer [JL97] presented a more efficient and easier to implement method for drawing evenly spaced streamlines in 2D space based on a neighboring seeding strategy. Verma et al. [VKP00] were interested in covering all topological features and proposed a method that places streamline seeds in special patterns according to the topological features for 2D flows. Ye [YKP05] recently extended this approach for 3D topological patterns. Mattausch et al. [MTHG03] used illuminated, evenly spaced streamlines with additional color mapping for visualizing 3D flow vector data. Another method for efficient streamline seeding for 2D fields, the "farthest-point streamline seeding," appropriate for optimizing continuity, was published by Mebarki et al. [MAD05]. Except from the work of Turk and Banks all these methods are based on an inter-sample distance control of the streamlines to reach a given density. Recently Liu et al. [LMG06] proposed a method, which improves robustness and efficiency of these distance based methods

## 2. Definition of the Streamline Density

As we want to control the density of streamlines, the first issue is to explain our understanding of streamline density. A streamline with a finite, non-zero line width and length covers a certain area when it is drawn. In the following we will call this area *streamline region*. Given a domain  $\Omega$  and a set of streamline regions  $R_S \subseteq \Omega$ , the global streamline density  $D_g$  is defined as:

$$D_g = \frac{\int_{\Omega} \chi_{R_S} d\Omega}{\int_{\Omega} 1 d\Omega}, \quad (1)$$

where  $\chi_{R_S}$  is the characteristic function of the subset  $R_S \subseteq \Omega$ .

It is reasonable to use a discrete representation, since in practice streamlines are drawn on a grid with certain resolution. For simplicity we assume that each cell only takes values of 0 or 1. Let the domain  $P$  be a set of cells, in 2D pixels or in 3D voxels, and  $S \subseteq P$  be the subset of streamline cells. The global discrete streamline density  $D_{gd}$

is defined as:

$$D_{gd} = \frac{\sum_{p \in P} \chi_S(p)}{|P|}, \quad (2)$$

where  $\chi_S$  is the characteristic function of the subset  $S \subseteq P$  and  $|P|$  the total number of cells.

This definition is quite instructive, since the number of occupied cells divided by the total number of cells indicates, in a range of  $[0, 1]$ , how dense the streamlines are.

Another interesting characteristic is the local streamline density. A certain density can be achieved by local windowing over the domain. Therefore, the discrete local streamline density  $D_{ld}$  in a cell represented by  $x_i$  is given as:

$$D_{ld}(x_i) = \sum_{p \in P} G(p - x_i) \chi_S(p), \quad (3)$$

where  $G(x)$  is a discrete filter with  $\sum_{p \in P} G(p - x_i) = 1$  and

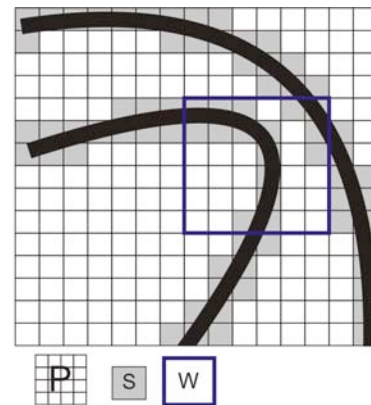
$W = \{p | G(p) \neq 0\} \subset P$  is a finite window. In our algorithm the local density is approximated using a Gauss-like filter defined over a square respective cubic domain, see Section 5. Figure 1 illustrates our streamline density definition.

## 3. Priority Streamline Algorithm

### 3.1. General Idea

The main idea behind our algorithm is to insert streamlines one by one into the visualization, where the seeding is guided by comparing a blurred version of the current image with the target density. In the following we first sketch our streamline generation algorithm works.

First, a so-called density map is constructed. According



**Figure 1:** Discrete streamline density definition. Our definition of density is based on the ratio of counted streamline pixels or voxels to the total number of pixels or voxels on the whole domain  $P$  or on a local domain  $W$ .

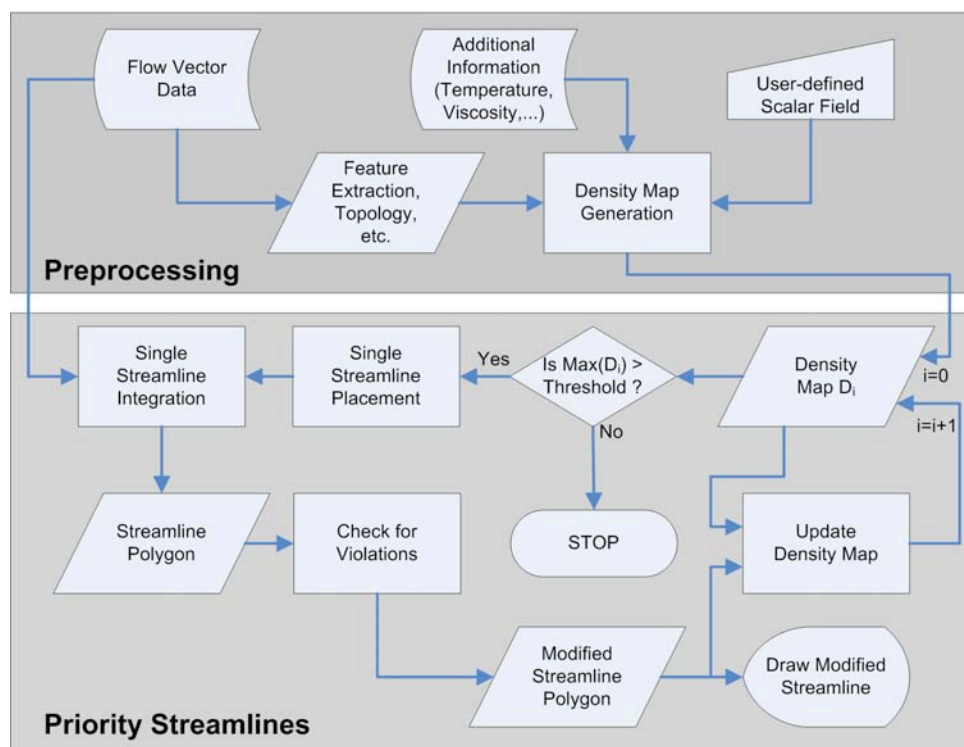


Figure 2: Flow diagram of priority streamline algorithm.

to this map streamline start points are seeded (mainly depending on the maxima of the map). The generation of each streamline lowers the density map locally until the map's global maximum is below a certain threshold. If this threshold is reached, the final image is ready. The algorithm will terminate in any case, as the density map is strictly monotonic decreasing over time. Figure 2 illustrates the algorithm generally as a flow diagram. The following paragraphs of section 3 will explain the basic algorithm in more detail. Two major issues, the density map and the filtering, are further inspected in sections 4 and 5.

### 3.2. Density Map

Given a flow vector data set, we want to draw streamlines with a density that is defined by a scalar function defined on the domain of the flow vector data set. This *density map* can be derived in many ways:

- *Definition considering additional data dimensions:* Application scientists often have more information than pure vector data. Temperature, viscosity, density, color, granularity, etc. could also be taken into account and define this map.
- *Definition considering derived vector information:* Given a vector field, one can calculate, for example, velocity and vorticity, with no additional information needed

for their computation. The topology of the field could be computed and serve as basis for the density map. Another way of deriving it from the field might consider pattern recognition algorithms, see [HEWK03, ES03, Sch04].

- *Definition by user:*

In our system, a user can define points and regions of interest by drawing the 2D or 3D density function manually. Moreover, the user can use pre-defined density maps and further edit them.

In our algorithm, the streamline density map is updated after each single streamline calculation, having an effect on further seeding positions as well as stop criteria. The construction of the density map is discussed in section 4.

### 3.3. Streamline Seeding

For the seeding of streamlines we decided to simply start streamlines at the current point of interest. This point is defined by the current streamline density map. For the first streamline, this is the maximum value of our initial density map. One special case must be considered where this principle does not really work. If we assume a constant density map, start points would be picked in a row. To prevent this, we add a small amount of noise to our initial density map, not influencing a non-uniform distribution. As the map is updated after each step to prevent too-close seeding, we

can pick the next maximum to seed the next streamline. In order to fulfill the continuity criterion, we decided to calculate distances of the next five maximum positions to our last streamline start point and take the farthest of these as current maximum. Another option would be to fully apply the farthest streamline seeding of Mebarki et al. [MAD05]. Our strategy works well for our purposes.

### 3.4. Calculation of the Streamlines

Given the streamline start position we integrate a full streamline in both directions. The general idea is to subtract a blurred, rasterized version of the streamline from the corresponding values of our density map. This can be done basically in two ways:

1. One can draw the streamline into a binary image, convolve it with a Gauss-like filter and subtract this image from the given density map.
2. One can traverse the streamline and subtract a Gauss-like filter kernel at each position consecutively from the density map.

Method 1 can be enhanced by calculating the convolution in frequency domain by using the fast Fourier transform. This would lead to a complexity of  $O(n \log_2 n)$ , with  $n = |P|$  being the total number of map pixels/voxels. This is in general too high a cost when processing 3D data. Method 2, referred to as *traversal algorithm*, turned out to be much more efficient. As the filter size is fixed, we obtain a linear time-complexity  $O(m)$ , with the number  $m = |S|$  of streamline pixels/voxels being much smaller than the total number of pixels/voxels. For this reason, this method is more suitable to apply our filter function. In detail, the polygon vertices of the streamline are mapped into the corresponding discrete density map space. There, from the start point, the Bresenham line drawing algorithm, see [Bre65], is applied to each of the polygon segments in both integration directions, to find the centers of the filter for each step. The Bresenham algorithm is a fast and robust algorithm for addressing each pixel/voxel of this mapped streamline.

The update process is split into two phases:

1. Checking for a violation
2. Final update of density map and drawing of streamline

In the first phase, the traversal algorithm checks whether a violation of the density map occurs when drawing the streamline. If the subtraction of the filter function results in negative values at any place of the map, the traversal algorithm stops and the last valid streamline pixel/voxel is stored (for both directions). At this point, we know both streamline ends. In phase two, these are mapped back into the original domain. The possibly shortened streamline is drawn on the screen, and the map is updated by a second application of the traversal algorithm.

## 4. Construction of the Density Map

For the construction of the density map we need a scalar field defined in the domain of the flow vector field. This scalar field can be given by additional data, user-defined, extracted as derived information from the vector field, or a combination of these (section 3.2).

In the first step, the scalar function  $f$  is discretized onto a fine regular grid, the so-called map. The resolution used for this discretization can be chosen according to the desired minimal streamline distance.

We require the density map to be non-negative. Depending on the application, this can be achieved in three ways: by computing the absolute value, by shifting, or by setting the negative values to zero.

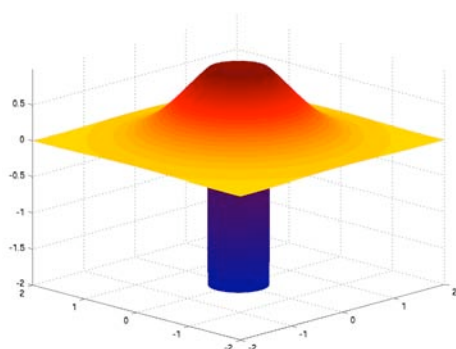
As we use a specific class of filters (see section 5 for details) we have to scale the density map to obtain appropriate results. The choice of the scale factor determines the general density of the drawn streamlines. We define the minimum height of our density function to have value 1.0. The maximum value of the function is user-defined and determines the streamline density, as well as the degree of importance. Let  $[a, b]$  be the range interval of the input scalar function. We define the goal range interval of our density map as  $[1, c]$ , with  $c$  being the so-called *importance factor*. We can now construct a linear map  $f : [a, b] \rightarrow [1, c]$ :

$$f(x) = \begin{cases} \frac{(x-a)(c-1)}{b-a} + 1, & \text{if } a \neq b \\ 1, & \text{if } a = b \end{cases} \quad (4)$$

The importance factor  $c$  can be selected by the user. If the factor is chosen near 1.0, the distribution of the streamlines is rather homogeneous. Increasing it yields an increasingly heterogeneous distribution according to the underlying importance map.

## 5. Filtering

In section 3.2 we explained that the density map is lowered frequently by a Gauss-like filter. We have chosen a Gaussian filter kernel since it is isotropic in all dimensions and allows us to produce a smooth transition between focused parts and edge regions. For our filter we need to incorporate a minimum distance. We do not want any streamline to lie in a certain  $\epsilon$ -region around the streamlines. One approach to solve this problem can be to define a second map, where those regions are marked separately. This can also be done in a simpler, more efficient way by using a trick in our filter design. We start with a filter similar to a Gauss filter. The major difference is the center region. There, we fill in a large negative value (ideally  $-\infty$ ). As we subtract the filter from the map we obtain a very high value in the map. For further streamline placement we exclude points of value higher than  $c$ . Thus, no streamline is placed in the forbidden



**Figure 3:** Constructed Gauss-like filter for 2D-data. The center region is set to a large negative value.

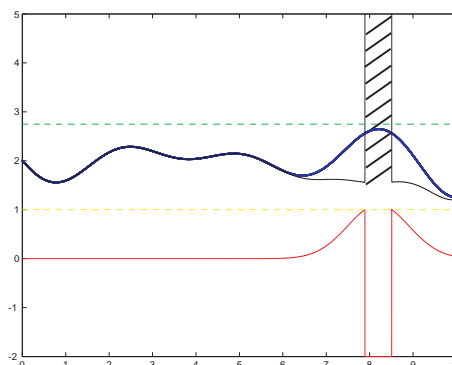
regions. We also include an additional check in our violation checking traversal algorithm. If there is a higher value than  $c$  at the current position, it lies in a forbidden  $\epsilon$ -region and the streamline is stopped. With these additional steps we manage to define and control the minimal distance.

Another difference to the Gauss filter is its maximum value and its sum. Usually, a Gauss function is normalized to have integral value one. In contrast to that, we define our filter to have value one as maximum and do not consider the integral. Of course, with a global definition one could control the number of streamlines by choosing a global scaling using a density map with a given volume and subtracting the Gauss filter volume frequently along the streamlines. But this turns out to be very complicated. For example, it is necessary to estimate the final streamline length in advance. Therefore, we have chosen the simpler local approach for the definition of our density map and our filter. To have a common basis, we define the maximum value of the Gauss-like filter to be one. Figure 3 shows a representation of the constructed filter. The final filtering process is also illustrated for a function in Figure 4.

## 6. Results

We have applied our method to numerically simulated data as well as simple synthetic data for test purposes. We have tested our method for 2D and 3D data. Our implementation was done in Python and embedded into the visualization tool "CoVE" (current project, University of Kaiserslautern). Our results have been generated on an Intel Centrino M 1.5 GHz, 512 MB RAM notebook.

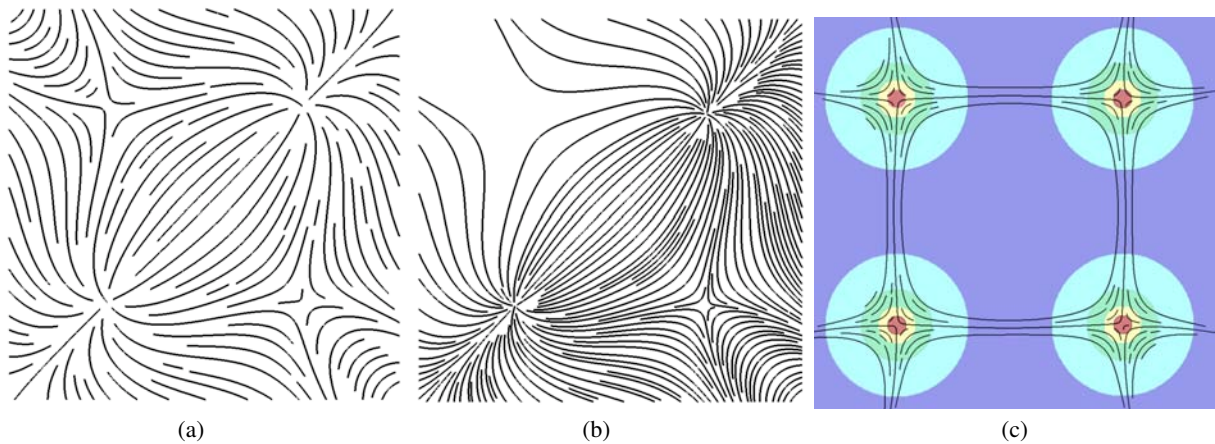
First, an artificial vector field is discussed, see Figure 5. It consists of  $512 \times 512$  vectors and represents two saddles, one sink and one source respective four saddles. For these vector fields, we used three different density maps. The first one was chosen completely homogeneously to compare our algorithm with standard algorithms. The second one was



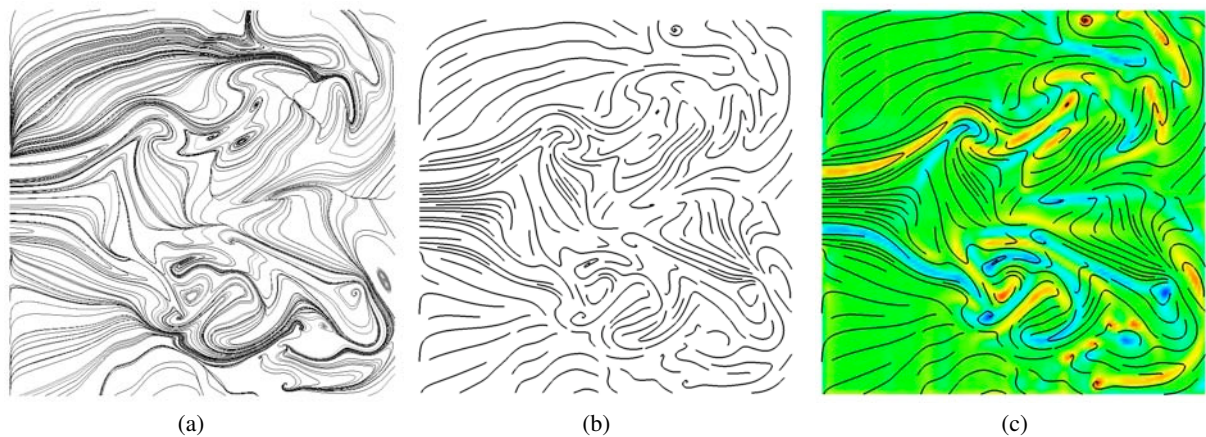
**Figure 4:** Filtering process illustrated for 1D-case. The density function (blue) is at its global maximum (below the green dotted line) subtracted by the filter (red). The resulting function (black) acts as density function for the next iteration. The hatched area shows the minimal distance zone of a streamline, where further placement is forbidden. The process is repeated until the global maximum of the current density function (below the green-dotted line) also lies below a certain threshold (here, the yellow dotted line).

chosen to have a high-valued center region and zeros on the edges, to show that we can use our algorithm for windowing purposes as well as for representing features through change of streamline density. The third density map highlights the saddle points. We applied the algorithm using various parameters. Adjusting the resolution of the density map and filter size directly affects overall streamline density. In section 4, we mentioned the so-called importance factor  $c$ . By increasing this factor, we increase the influence of the underlying density and introduce (depending on the given density map) a higher degree of heterogeneity, see Figure 5(b).

As an example for non-trivial 2D data we decided to use a slice of a simulated data set representing a swirling jet entering a fluid at rest. The data is given on a rectilinear, non-uniform grid. Figures 7(a) and (b) show the data imaged with standard streamline generators, compared to images generated with our algorithm using the velocity field of the data as density map. One can see that the standard streamlines do not provide any information on velocity, while our approach reveals the velocity through streamline density. Other visual mappings like color mapping can be used to visualize and compare other properties of the data. As an example, Figure 7(c) shows the priority-on-high-velocity streamlines together with a scalar color mapping representing vorticity. One can see that there is a relation between vorticity and velocity. In regions of high vorticity there is lower velocity. As this is a known fact, it confirms the validity of our method. Other visualization parameters like transparency, illumination, varying streamline thickness



**Figure 5:** Two test datasets: (a) and (b) two saddles, one source and one sink, (c) four saddles and one center point. (a) Streamlines drawn by the priority streamline algorithm using (a) a constant density map, (b) a heterogeneous density map: values of density map increase from the upper left to the lower right corner. (c) a density map highlighting saddle points. The density map is shown as background color.



**Figure 7:** (a) Swirling jet entering fluid at rest. Visualized with standard streamlines. (b) Swirling jet data visualized with priority streamlines using velocity magnitude as density map. (c) Comparative visualization of velocity (streamline density) and vorticity of swirling jet data.

(streamtube thickness), etc. could be used for mapping additional properties.

Finally, we applied our algorithm to 3D data. First we apply our algorithm to a flow vector data set from a simulation of convection in the Earth mantle. We choose the density map in a way that our method acts as a filter and represents only streamlines in desired regions. The desired regions can be chosen manually (through editing the 3D density map) or by mapping any other property of the data to the density map. In our example we considered density maps based on velocity, temperature and different criterions for vortex region detection: vorticity and

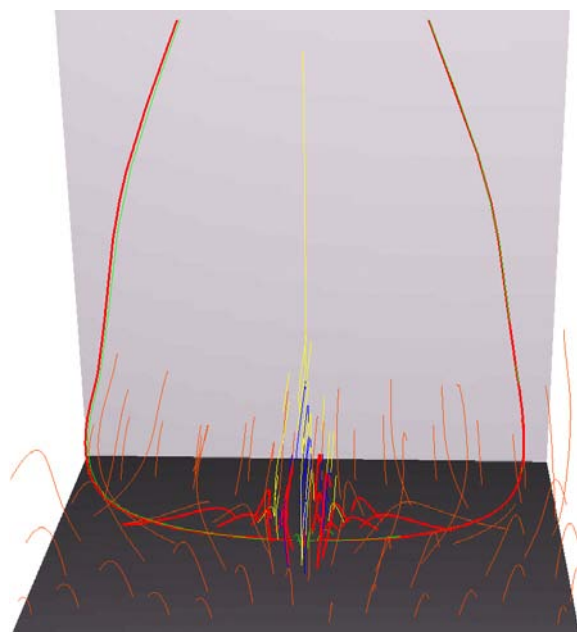
Okubo-Weiss (OW) parameter, for more details see Sahner et al. [SWH05]. Figure 6 shows all streamline types using different colors. One can see the basic flow behavior as well as other important properties (e.g., vorticity, velocity, and temperature distribution) in just one image. Clutter, a major issue in visualizing streamlines in three dimensions, is reduced by our algorithm. The second example is a snapshot of the velocity field of a 3D mixing layer. Figure 8 shows three different views onto the result generated by our method using various density maps together with an vorticity isosurface. The representation of scalar fields using streamline density makes it possible to display several fields at a time without occluding each other. This allows a

side by side comparison of these fields, e.g., two different parameters for vortex detection.

## 7. Conclusions

We have presented a new algorithm for drawing streamlines with a defined heterogeneous density. It is useful for various tasks. Priority streamlines can be used to present additional properties beyond vector data (like temperature), to represent implicitly given but not visualized data (e.g., velocity, vorticity, etc.), to compare or to filter (e.g., to reduce clutter in 3D visualizations). Priority streamlines can also be used to compute homogeneously distributed streamlines. The results for homogeneous distributions are not as good, as those obtained with special solution approaches, focussing on homogeneity. We do not see our algorithm in competition with these techniques, as our goals are different. We skip the *continuity* criterion to obtain a higher degree of freedom for the representation of different streamline densities and filtering purposes. We have implemented an efficient method using a pre-processing step to decrease computation time during user interaction. Our results show that we are able to control our visualization in ways to highlight and extract certain information. Control parameters can be chosen manually or be derived automatically.

Future research will be directed at the choice and con-



**Figure 6:** Convection in Earth mantle. Red: streamlines indicate high vorticity, yellow: high velocity, green: high (positive) values for Okubo-Weiss (OW), blue highly negative values for OW, orange: high temperature.

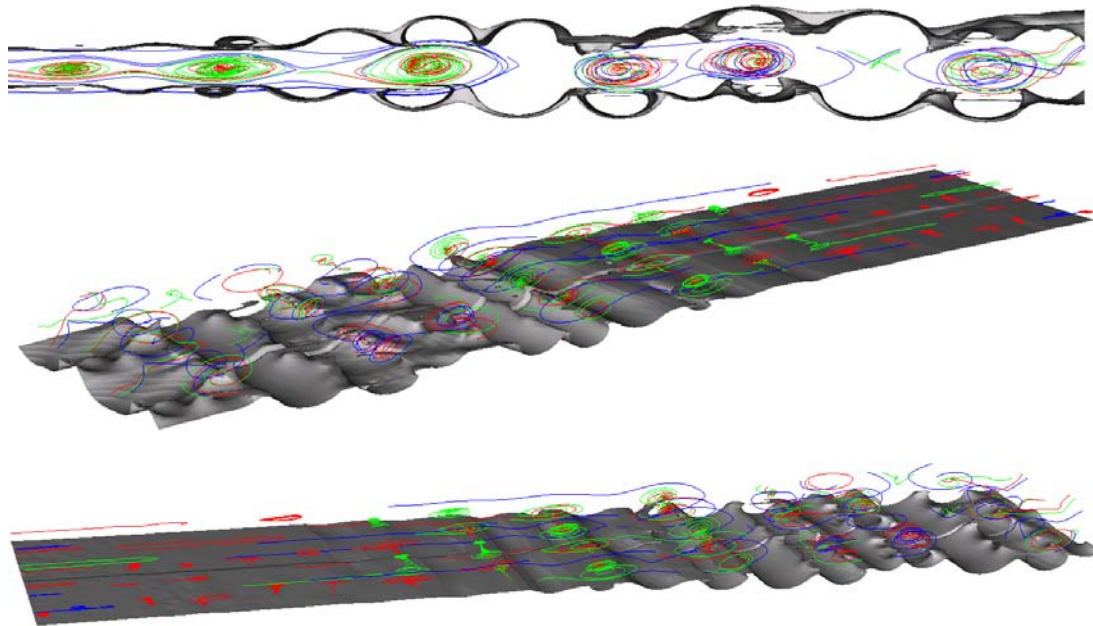
struction of density maps. The right choice of a density map can strongly influence and enhance a visualization. Smoothing of sharp density maps might lead to a more pleasant representation, as streamline length will probably increase. It should also be analyzed how priority streamlines behave in combination with additional visual modifications (e.g., transparency).

## 8. Acknowledgments

We thank Wolfgang Kollmann, Department of Mechanical and Aeronautical Engineering, Louise Kellogg, Department of Geology, University of California, Davis, and Pierre Comte, Institut de Mecanique des Fluides et des Solides, University Louis Pasteur Strasbourg, France for providing the simulated data sets. We express our gratitude to the members of the Visualization and Computer Graphics Research Groups at the University of Kaiserslautern and the Institute for Data Analysis and Visualization (IDAV) at the University of California, Davis, as well as the members of the International Research Training Group (IRTG), especially Christoph Garth and Manuel Heringer. The IRTG is supported by the German Research Foundation (DFG). This work was also supported in part by the National Science Foundation under contract ACI 9624034 (CAREER Award) and a large Information Technology Research (ITR) grant. We gratefully acknowledge the support of the W.M. Keck Foundation provided to the UC Davis Center for Active Visualization in the Earth Sciences (CAVES).

## References

- [Bre65] BRESENHAM J.: Algorithm for computer control of a digital plotter. *IBM Systems Journal* 4, 1 (1965), 25–30.
- [ES03] EBLING J., SCHEUERMANN G.: Clifford convolution and pattern matching on vector fields. In *Proceedings of IEEE Visualization 2003* (Los Alamitos, CA, USA, 2003), Turk G., van Wijk J. J., Moorhead R., (Eds.), IEEE Computer Society Press, pp. 193–200.
- [HEWK03] HEIBERG E., EBBERS T., WIGSTRÖM L., KARLSSON M.: Three-dimensional flow characterization using vector pattern matching. *IEEE Transactions on Visualization and Computer Graphics* 9, 3 (2003), 313–319.
- [JL97] JOBARD B., LEFER W.: Creating evenly-spaced streamlines of arbitrary density. In *Visualization in Scientific Computing '97. Proceedings of the Eurographics Workshop in Boulogne-sur-Mer, France* (Wien, New York, 1997), Lefer W., Grave M., (Eds.), Springer Verlag, pp. 43–56.
- [LMG06] LIU Z., MOORHEAD R., GRONER J.: An advanced evenly-spaced streamline placement algorithm. *IEEE Transactions on Visualization and Computer Graphics* 12, 5 (2006), 965–972.



**Figure 8:** Snapshot of the velocity field of a 3D mixing layer. Three views onto the dataset. Red: streamlines indicate high vorticity, yellow: high velocity, green: high (positive) values for OW, blue highly negative values for OW.

- [MAD05] MEBARKI A., ALLIEZ P., DEVILLERS O.: Farthest point seeding for efficient placement of streamlines. In *VIS '05: Proceedings IEEE Visualization '05* (Los Alamitos, CA, USA, 2005), Silva C. T., Gröllner E., Rushmeier H., (Eds.), IEEE Computer Society Press, pp. 479–486.
- [MHHI98] MAO X., HATANAKA Y., HIGASHIDA H., IMAMIYA A.: Image-guided streamline placement on curvilinear grid surfaces. In *VIS '98: Proceedings of the conference on Visualization '98* (Los Alamitos, CA, USA, 1998), Ebert D., Rushmeier H., Hagen H., (Eds.), IEEE Computer Society Press, pp. 135–142.
- [MTHG03] MATTAUSCH O., THEUSSL T., HAUSER H., GRÖLLER E.: Strategies for interactive exploration of 3d flow using evenly-spaced illuminated streamlines. In *SCCG '03: Proceedings of the 19th spring conference on Computer graphics* (New York, NY, USA, 2003), ACM Press, pp. 213–222.
- [Sch04] SCHLEMMER M.: *Fourier Transformation and Filter Design for Clifford Convolution*. Master's thesis, University of Kaiserslautern, 2004.
- [SWH05] SAHNER J., WEINKAUF T., HEGE H.-C.: Galilean invariant extraction and iconic representation of vortex core lines. In *Proc. Eurographics / IEEE VGTC Symposium on Visualization (EuroVis '05)* (Leeds, UK, June 2005), K. Brodli D. Duke K. J., (Ed.), pp. 151–160.
- [SWHS97] SALISBURY M. P., WONG M. T., HUGHES J. F., SALESIN D. H.: Orientable textures for image-based pen-and-ink illustration. *Computer Graphics 31*, Annual Conference Series (1997), 401–406.
- [TB96] TURK G., BANKS D.: Image-guided streamline placement. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM Press, pp. 453–460.
- [VKP00] VERMA V., KAO D., PANG A.: A flow-guided streamline seeding strategy. In *VIS '00: Proceedings of the conference on Visualization '00* (Los Alamitos, CA, USA, 2000), Ertl T., Hamann B., Varshney A., (Eds.), IEEE Computer Society Press, pp. 163–170.
- [YKP05] YE X., KAO D., PANG A.: Strategy for seeding 3d streamlines. In *VIS '05: Proceedings IEEE Visualization '05* (Los Alamitos, CA, USA, 2005), Silva C. T., Gröllner E., Rushmeier H., (Eds.), IEEE Computer Society Press, pp. 471–478.