Cusps of Characteristic Curves and Intersection-Aware Visualization of Path and Streak Lines

Tino Weinkauf, Holger Theisel, Olga Sorkine

Abstract We analyze characteristic curves of vector fields and report on locations where they have cusps in their spatial projection, i.e., isolated points on the curve with abruptly turning tangent direction. Cusps appear in places where a projection of the corresponding tangent curve vector field exhibits critical points. We show that such cusps are only possible for streak and path lines, whereas they cannot appear on stream and time lines. Cusps turn out to be closely related to self-intersections of characteristic curves. We utilize this information in a new algorithm to create uncluttered static visualizations of path and streak lines.

1 Introduction

Dynamic flow phenomena play an important role in many applications. In this paper, we are particularly interested in 2D time-dependent flows. Their three-dimensional space-time domain allows essentially for the following visualization options:

- **Static space-time visualization:** Time is explicitly incorporated as the third dimension and the geometry or texture to be visualized is shown in the space-time volume. Examples are tracked critical points shown as line-type structures in space-time [11, 12], or path lines stretching through the volume (cf. Figure 2b).
- **Dynamic visualization in space:** The temporal behavior is captured in an animation of a 2D visualization. Examples are particle animations [4], animations of FTLE color plots [2], animated texture-based flow visualizations [15, 17], and many more.

Tino Weinkauf and Olga Sorkine
Courant Institute of Mathematical Sciences, New York University, 715 Broadway, New York, NY 10003, U.S.A., e-mail: {weinkauf, sorkine}@courant.nyu.edu

Holger Theisel
AG Visual Computing, Otto-von-Guericke-Universität, Universitätsplatz 2, 39106 Magdeburg, Germany, e-mail: theisel@isg.cs.uni-magdeburg.de

- **Static visualization in space:** The temporal behavior is encoded in the 2D visualization by means of color or other graphical attributes. An example is the visualization of path lines where time can be mapped to the width of the line. An example for 3D flows is given by Wiebel and Scheuermann [16], where bundles of path and streak lines running through the same spatial location are visualized in a static image.

Static visualizations not only have the advantage of being perfectly suited for non-dynamic types of media such as paper, but they also compile all information into a single image, whereas animations break it into several transient impressions. Clearly, both the dynamic and the static approach have their merits.

In this paper we are concerned with static visualizations of characteristic curves (stream, path, streak, time lines; Section 2). While visualizing them in 3D space-time is always a valid option, note that this boils down to a 2D projection anyway once a snapshot is taken for purposes such as printing, and the non-orthogonal, perspective projection inevitably introduces distortions as showcased throughout the paper. In contrast, visualizing characteristic curves in the spatial domain (by means of an orthogonal, orthographic projection) leaves the physically meaningful spatial domain undistorted and the temporal dimension can be encoded in graphical attributes.

Various stream line placement methods are available that aim at expressive visualizations by distributing stream lines (of a time step or a steady flow) in an evenly-spaced manner [3, 6, 8, 13], or for 3D steady flows in a view-dependent fashion to reduce visual clutter [5]. To the best of our knowledge and much to our surprise, there seem to be no existing methods for the placement of the other three types of characteristic curves in time-dependent flows. There are many texture- or geometry-based approaches for path, streak and time lines in general [1, 9, 15], but none of them take care of the distances between lines or possible intersections.

In this paper, we develop a method to create uncluttered, static visualizations of path and streak lines. To that end, we first analyze the challenges that come with the purely spatial depiction of these curves. Besides their obvious intersections in space, we pay special attention to cusps (isolated points on the curve with abruptly turning tangent direction) and self-intersections (Sections 3 and 4). It turns out that both phenomena are closely related to the occurrence of critical points in the original flow. Based on the gained insight into the intricacies of visualizing path and streak lines in space, we design a placement algorithm for them that makes use of the flow topology and aims at cusp-free and intersection-aware visualizations (Section 5). As we will show in our results (Section 6), an *entirely* intersection-free visualization is of limited use for turbulent flows. We therefore allow for a small number of (self)-intersecting path or streak lines.

Notation. We consider a 2D time-dependent vector field $\mathbf{v}(\mathbf{x},t)$ over the domain $D \times T$ where $D \subseteq \mathbb{R}^2$ is the spatial domain and T is a time interval. We write $(2+1)$-dimensional variables with a bar like $\bar{\mathbf{p}}$, and $(2+2)$-dimensional variables with a double bar like $\bar{\bar{\mathbf{q}}}$.

2 Characteristic Curves of Vector Fields

A curve L is called a *tangent curve* of a vector field $\mathbf{v}(\mathbf{x})$, if for all points $\mathbf{p} \in L$ the tangent vector of L coincides with $\mathbf{v}(\mathbf{p})$.

In a time-dependent vector field $\mathbf{v}(\mathbf{x},t)$ there are four types of characteristic curves: stream lines, path lines, streak lines and time lines. We can start a *stream line* in a space-time point $(\mathbf{x}_0,t_0)$, staying in time slice $t = t_0$, by integrating a tangent curve in the vector field $\bar{\mathbf{s}}(\mathbf{x},t) = (\mathbf{v}(\mathbf{x},t),0)^T$. Similarly, *path lines* of the original vector field $\mathbf{v}$ are described as the tangent curves of the vector field $\bar{\mathbf{p}}(\mathbf{x},t) = (\mathbf{v}(\mathbf{x},t),1)^T$ in space-time. Path lines describe the trajectories of massless particles in time-dependent vector fields.

A *streak line* is the collection of all particles set out at different times but the same point location. In an experiment, one can observe these structures by constantly releasing dye into the flow from a fixed position. The resulting streak line consists of all particles which have been at this fixed position sometime in the past. Considering the vector field $\bar{\mathbf{p}}$ introduced above, streak lines can be obtained in the following way: apply a path surface integration in $\bar{\mathbf{p}}$ where the seeding curve is a straight line segment parallel to the t-axis; a streak line is the intersection of this path surface with a hyperplane perpendicular to the t-axis. As shown in [14], streak lines can be described as tangent curves of the vector field

$$\bar{\bar{\mathbf{q}}}(\mathbf{x},t,\tau) = \begin{pmatrix} (\nabla\phi_t^\tau(\mathbf{x}))^{-1} \cdot \frac{\partial\phi_t^\tau(\mathbf{x})}{\partial t} + \mathbf{v}(\mathbf{x},t) \\ 0 \\ -1 \end{pmatrix}, \tag{1}$$

where $\phi_t^\tau(\mathbf{x})$ denotes the flow map computed from particles seeded at $(\mathbf{x},t)$ and integrated over a time interval τ. We call $\bar{\bar{\mathbf{q}}}$ the *streak line vector field*. It is defined in the domain $D \times T \times \Upsilon$ with $\tau \in \Upsilon$, i.e., $\bar{\bar{\mathbf{q}}}$ is a 4D vector field if the original flow $\mathbf{v}$ is a 2D time-dependent field. The streak lines of a constant time step can be integrated as tangent curves in the subspace $D \times \Upsilon$, i.e., $\bar{\bar{\mathbf{q}}}$ simplifies to

$$\bar{\mathbf{q}}'(\mathbf{x},\tau) = \begin{pmatrix} \mathbf{w}(\mathbf{x},\tau) \\ -1 \end{pmatrix}, \qquad t = \text{const}, \tag{2}$$

where $\mathbf{w}(\mathbf{x},\tau) = (\nabla\phi_t^\tau(\mathbf{x}))^{-1} \cdot \frac{\partial\phi_t^\tau(\mathbf{x})}{\partial t} + \mathbf{v}(\mathbf{x},t)$ denotes the spatial components.

A *time line* is the collection of all particles set out at the same time but different locations, i.e., a line which gets advected by the flow. An analogon in the real world is a yarn or wire thrown into a river, which gets transported and deformed by the flow. However, in contrast to the yarn, a time line can get shorter and longer. It can be obtained by applying a path surface integration in $\bar{\mathbf{p}}$ starting at a line with $t = \text{const}$, and intersecting it with a hyperplane perpendicular to the t-axis. Whether time lines can be described as tangent curves of some derived vector field is still an open research question.

See [14] for a more thorough discussion of characteristic curves.

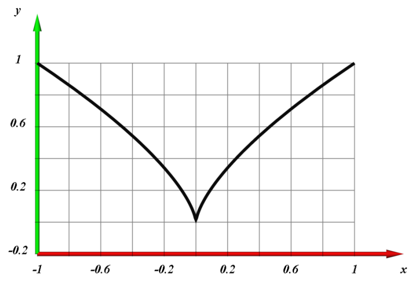


Fig. 1 Plot of the function $x^2 - y^3 = 0$. It has a cusp at the origin, where the tangent vector of the curve abruptly changes its direction.

3 Cusps of Characteristic Curves

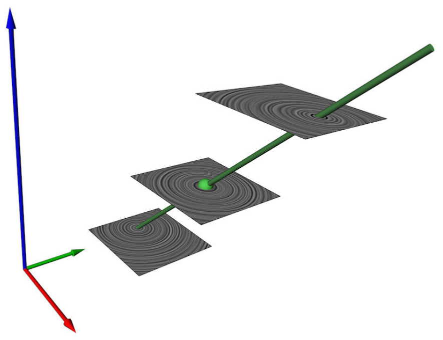
Cusps are isolated points on a curve where its tangent vector abruptly changes, i.e., the tangent has a discontinuity. Figure 1 shows this for a simple example: the zero-levelset of $x^2 - y^3$, which has a cusp at the origin.

The previously introduced characteristic curves can be described as tangent curves (except for time lines) in their higher-dimensional domains $D \times T$ or $D \times T \times \Upsilon$. Given that the respective tangent curve vector fields $\bar{\mathbf{s}}, \bar{\mathbf{p}}, \bar{\bar{\mathbf{q}}}$ are continuous, their tangent curves are smooth, i.e., they *cannot* have cusps in these higher-dimensional spaces. This follows since the vector field gives the first derivative of the tangent curve: if the vector field is continuous (C^0), then the first derivative of the tangent curve is C^0, which makes the tangent curve itself smooth (C^1).

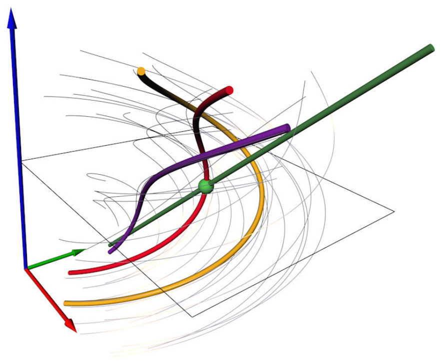
However, a projection of a characteristic curve into a subspace may very well have a cusp. In this paper, we are interested in a *spatial* projection, since we wish to create static, two-dimensional visualizations of these curves in Section 5. We strive for high expressiveness of these visualizations by reducing the amount of clutter that is likely to be introduced by the projection. To that end, we analyze the occurrence of cusps in these projections, since the abrupt turning of the tangent direction at cusps communicates a non-smoothness to the viewer despite the fact that the underlying field is actually smooth or at least continuous. Hence, we want to exclude cusps (and nearby areas) from these visualizations. In the following, we study cusps in spatial projections of all four types of characteristic curves.

Stream Lines. The case for stream lines is rather trivial. Stream lines live in a constant time step, i.e., a spatial projection does not alter their geometry and they cannot have cusps in a continuous vector field.

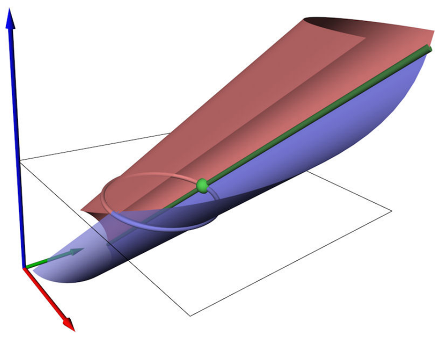
Time Lines. Although a tangent curve description for time lines is not yet available, we can already remark the following: a time line can be seen as the front of a path surface integration in $\bar{\mathbf{p}}$. Assuming that $\bar{\mathbf{p}}$ is smooth and remembering that it does not have any critical points, the front line undergoes only smooth transformations during the integration. Since t is constant for a time line, a spatial projection will not alter the geometry of the curve. Hence, time lines cannot have cusps. They may however be subject to strong bending during the integration, which may distort the region around some point in a way that resembles a cusp, but those points would not be cusps in the infinitesimal sense.



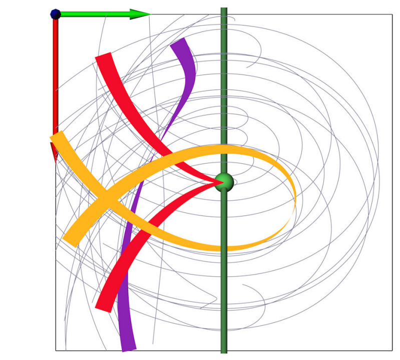
(a) Stream lines shown with LIC. Tracked critical point shown as straight green line.



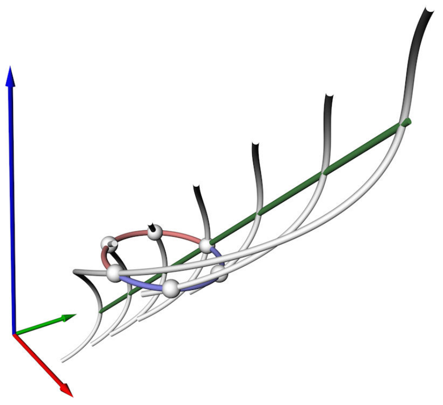
(b) 28 path lines, three of them highlighted. See below for their spatial projection.



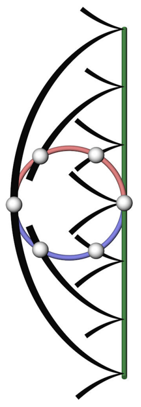
(c) Forward (red) and backward (blue) path surface started from the tracked critical point. Intersection with t_0 yields cusp seeding line.



(d) Spatial projection of the path lines from (b). Highlighted path lines are scaled by distance to t_0 (shown in (b) by the large quad).



(e) Every path line started on the cusp seeding line (red/blue) runs through the critical point.



(f) Spatial projection of (e). Path lines shown in black and scaled by distance to cusp.

Fig. 2 Cusps in the spatial projection of path lines shown for the 2D time-dependent vector field (3). The red and green axes denote the spatial (x, y)-domain, whereas the temporal dimension is depicted by the blue axis. The time step t_0 is denoted by the large quad in (b) and (c).

3.1 Cusps in the Spatial Projection of Path Lines

Path lines of $\mathbf{v}(\mathbf{x},t)$ are the tangent curves of $\bar{\mathbf{p}}$ in space-time. The first two components of $\bar{\mathbf{p}}$ become zero at locations $\mathbf{c}$ where $\mathbf{v}$ has a critical point, i.e., $\bar{\mathbf{p}}(\mathbf{c}) = (0,0,1)^T$. A path line running through $\mathbf{c}$ advances there only in temporal direction, but stays at the same spatial location (for an infinitesimally small time). Consequently, a following spatial projection will lead to a cusp at this point.

We can see this in Figure 2, where the vector field

$$\mathbf{v}(x,y,t) = \begin{pmatrix} -y \\ x-t \end{pmatrix} \tag{3}$$

is visualized. It contains a critical point of type *center* that moves over time in x-direction. Figure 2a shows its evolution as a straight green line in space-time together with LIC visualizations of the stream lines in three selected time steps. Note that the spatial dimensions in all following visualizations are depicted by the red and green axes, whereas time is denoted by the blue axis.

Figures 2b and 2d show a number of path lines in space-time and in the spatial projection, respectively. Three of them have been highlighted in pink, red, and yellow. We make the following observations for the spatial projection (Figure 2d):

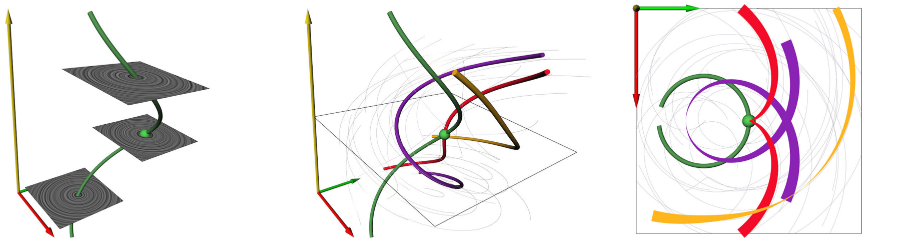
- A lot of visual clutter is apparent due to path lines intersecting each other.
- Path lines may have self-intersections as exemplified by the yellow curve.
- Path lines have cusps when they run through a critical point of $\mathbf{v}$ during their integration, as exemplified by the red curve. Note that this curve is perfectly smooth in space-time (Figure 2b).

We will deal with the (self-)intersection issues in Sections 4 and 5. Here it remains to understand which subset of all space-time seeding locations gives rise to path lines with cusps in their spatial projection. Later, we will exclude those areas from our seeding algorithm.

The critical points of $\mathbf{v}$ are line-type structures in space-time. They can be extracted with a number of methods, for example using Feature Flow Fields [10, 11]. Since all path lines with cusps have to intersect these critical lines, we can collect all their possible seeding locations using two path surface integrations in $\bar{\mathbf{p}}$ started from each critical line: one in forward and one in backward direction. Figure 2c illustrates this. Intersecting the surfaces with a time step t_0 yields *cusp seeding lines*: they describe all seeding locations at t_0 which give rise to path lines with cusps in their spatial projection. Note that depending on where we start a path line integration on a cusp seeding line, the actual crossing of the critical point might be before, after or at t_0. Figure 2e shows this in space-time, Figure 2f in space.

3.2 Cusps in the Spatial Projection of Streak Lines

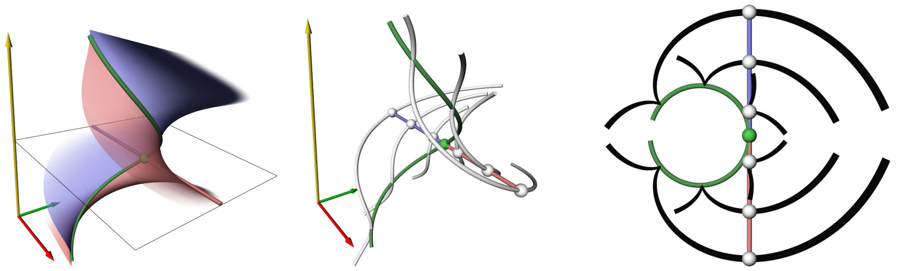
As described in Section 2, streak lines live in a constant time step and are given there by Equation (2) as the tangent curves of $\bar{\mathbf{q}}'$ in the 3D domain $D \times \Upsilon$, where Υ



(a) Spatial components of $\bar{\mathbf{q}}'$ shown with LIC. Tracked critical point shown as green line.

(b) 28 streak lines, three of them highlighted. See right for their spatial projection.

(c) Spatial projection of the streak lines from (b). Highlighted streak lines are scaled by distance to τ_0 (shown in (b) by the large quad).



(d) Forward (red) and backward (blue) integral surface in $\bar{\mathbf{q}}'$ started from tracked critical point. Intersection with τ_0 yields cusp seeding line.

(e) Every streak line started on the cusp seeding line (red/blue) runs through the critical point.

(f) Spatial projection of (e). Streak lines shown in black and scaled by distance to cusp.

Fig. 3 Cusps in the spatial projection of streak lines shown for the 2D time-dependent vector field (3). Shown is the (x, y, τ)-subspace (red, green, yellow axes) at $t = 0$, where streak lines are given as tangent curves of $\bar{\mathbf{q}}'$. The large quad in (b) and (c) denotes $\tau_0 = 0$.

refers to the τ-dimension denoting the integration interval. Hence, a projection that removes the τ-dimension is required regardless of whether we visualize streak lines in space-time or just space.

Such a projection leads to cusps, in a similar way as for path lines, at locations $\mathbf{c}$ where $\mathbf{w}$ has a critical point, i.e., $\bar{\mathbf{q}}'(\mathbf{c}) = (0, 0, -1)^T$. It can be shown that this implies $\mathbf{v}(\phi_t^\tau(\mathbf{x})) = \mathbf{0}$. In other words, the critical points of the original flow are closely related (via the flow map ϕ) to the critical points of the spatial components of the streak line vector field.

Figure 3 shows this for the example vector field (3), that we already used in the previous section. Its streak line vector field is given as $\bar{\bar{\mathbf{q}}}(x, y, t, \tau) = (\cos(\tau) - 1 - y, -\sin(\tau) + x - t, 0, -1)^T$. We show only the (x, y, τ)-subspace at $t = 0$, where streak lines are given as tangent curves of $\bar{\mathbf{q}}'(x, y, \tau) = (\cos(\tau) - 1 - y, -\sin(\tau) + x, -1)^T$.

The spatial components of $\bar{\mathbf{q}}'$ are visualized using LIC in Figure 3a together with the tracked critical point. Note the difference to the tracked critical point of $\mathbf{v}$ shown in Figure 2a.

Figures 3b and 3c show a number of streak lines in $D \times \Upsilon$ and in the spatial projection, respectively. The three highlighted streak lines exemplify, that we have to deal with (self-)intersections and cusps for streak lines as well.

We collect all possible seeding locations for streak lines with cusps similar to the path line case: integrate two surfaces (forward/backward) in $\bar{\mathbf{q}}'$ starting from every critical line (Figure 3d). Their intersection with a certain τ_0 gives the cusp seeding lines, that describe all seeding locations at τ_0 which give rise to streak lines with cusps in their spatial projection (Figures 3e-f).

4 Remarks on Self-Intersections of Characteristic Curves

For the sake of simplicity, the following remarks are made for path lines. They extend to streak lines in a similar manner.

Consider a path line seeded at $(\mathbf{x}, t_0)$ that exhibits a self-intersection in its spatial projection – as it is shown in the leftmost image of Figure 4. Assume that it is possible to change its spatial seeding location (staying in t_0) such that the loop created by the intersection becomes gradually smaller. Eventually, the loop will degenerate to a point: a cusp. The seeding location is now on a cusp seeding line ℓ_c and the previous seeding locations have been on one side of ℓ_c (in a local sense). Placing the seed on the other side of ℓ_c yields a path line without a self-intersection.

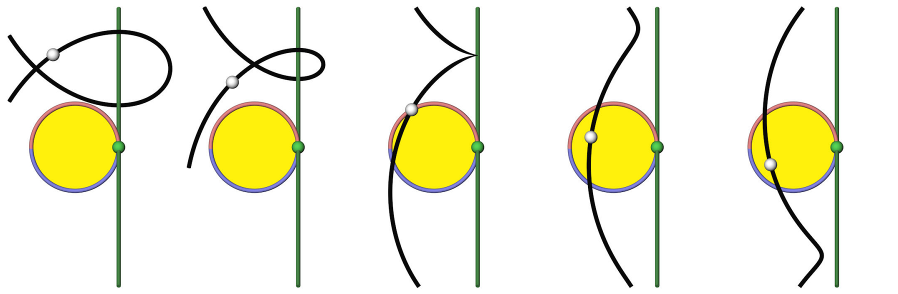
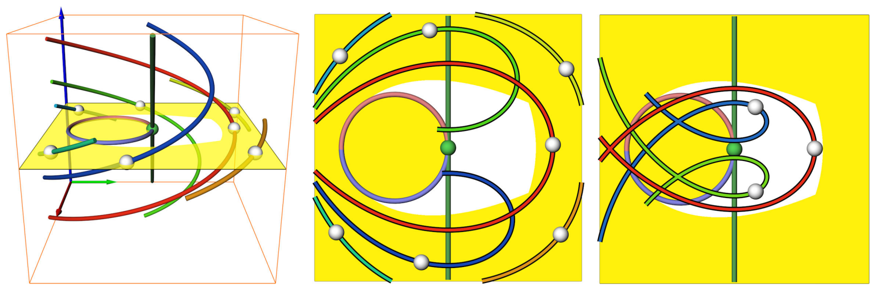


Fig. 4 Gradually changing the seeding location (gray ball) of a path line (black) leads to a qualitatively different behavior when crossing a cusp seeding line (red/blue curve). Shown is the example vector field (3), where only seeding locations inside the circle (yellow area) give rise to non-self-intersecting path lines.

It is easy to see, that cusp seeding lines play a vital role in the binary segmentation of a time step into areas where seeding a path line leads to self-intersection(s) and where it does not. Figure 4 illustrates this for our example vector field (3) from the previous section. Here, we have a single cusp seeding line in the shape of a circle. In general, cusp seeding lines are not closed and are of arbitrary shape. It turns out that all seeding locations outside the circle give rise to self-intersecting path lines, whereas only path lines seeded inside the circle do not have a self-intersection in their spatial projection.



(a) Path lines seeded where the boundary prematurely prevents their self-intersection.

(b) Spatial projection of (a).

(c) Only a rather small part of the domain is left to give rise to self-intersecting path lines.

Fig. 5 Defining a boundary $[-2.5, 2.5]^3$ for the example vector field (3) limits the area (white) where seeding path lines leads to their self-intersection in the spatial projection.

At least one other component contributes to the self-intersection segmentation of a time step: the boundary of the domain – may this be a boundary given as a hard constraint by a numerical simulation, or as a soft constraint defined by a user. A possible self-intersection of a path line is prevented if the integration stops at the boundary before having reached the intersection point. In other words, introducing a boundary *reduces* the number of self-intersecting path lines. Figure 5 illustrates this.

There is an equivalent to cusp seeding lines associated with the boundary: one could call them "boundary touching lines." They consist of seeding locations at the boundary giving rise to path lines, which have one end touching another part of the spatially projected path line. To one side of a "boundary touching line" we find self-intersecting path lines, to the other side the ones that could not make it to the self-intersection due to the premature integration stop at the boundary.

As a last remark, we also suspect topological separatrices (emanating from saddle points) to play a role in the self-intersection segmentation.

However, for the purposes of this paper, it suffices to understand that only a subset of all possible seeding locations (in some flows actually a rather small subset) gives rise to self-intersecting path lines. This will guide some algorithmic decisions in the following section.

5 Intersection-Aware Visualization of Path and Streak Lines

Our method for creating uncluttered, static visualizations of path or streak lines follows these principles:

- Only a very limited, predetermined number of field lines is allowed to have self-intersections and intersect each other. This is required to highlight turbulent areas.

Algorithm 1 Intersection-Free Placement of Field Lines *(Python-like syntax)*

```
def FieldLinePlacement(nDesiredLines):
    ResultLines = []
    Pool = IntegrateFieldLines(m) #Initialize with m randomly seeded field lines. default: m = 100

    while len(ResultLines) < nDesiredLines:
        Pool += IntegrateFieldLines(n) #Get n new randomly seeded field lines. default: n = 30
        Pool.SortByLength() #Sort field lines in descending order; longest comes first.
        LengthOfLastLongLine = Pool[0].Length()
        ResultLines += Pool.pop(0) #Add the currently longest line to the result.

        while len(ResultLines) < nDesiredLines:
            Pool.CutFieldLinesIntersectingWith(ResultLines) #Intersection-free!
            Pool.SortByLength() #Sort field lines in ascending order; longest comes first.
            if Pool[0].Length() / LengthOfLastLongLine > x: #default: x = 0.98
                ResultLines += Pool.pop(0) #Add the currently longest line to the result.
            else:
                break #Available field lines got too small. Get new ones.

        #Delete small lines from the pool before we add new ones in the next loop.
        Pool.RemoveLinesSmallerThan(LengthOfLastLongLine * y) #default: y = 0.5

    return ResultLines
```

- All other field lines shall not have any intersections.
- Long field lines are preferred.
- A good coverage of the domain is desired.
- Field lines shall have a certain minimal distance to each other.
- Cusps and cusp-like shapes shall be avoided.

The main ingredient of our method is Algorithm 1, which achieves a completely intersection-free placement of field lines with a reasonable domain coverage and preference for long lines. The basic idea is to randomly seed a large number of field lines, put them in a pool of available lines, and iteratively copy the longest one into the result. After adding a new line to the result, all remaining lines in the pool have to be shortened such that they do not intersect one of the result lines. We continue with the process of adding the currently longest line to the result and shortening the remaining ones in the pool until the lines in the pool become too short. Then we add a number of new field lines to the pool and continue to do so until we have reached a desired number of field lines in the result.

We use a texture-based approach to check for intersections: all field lines in the result are also rendered into a 2D texture. Testing a field line from the pool for intersection with the result lines is then just a matter of checking for overlapping pixels. The texture also allows us to maintain a certain minimal distance between lines by using a certain pixel width when rendering new lines into the texture. This works nicely together with the intersection test and also with the seeding of new field lines, where spatial locations with covered pixels are excluded from the seeding.

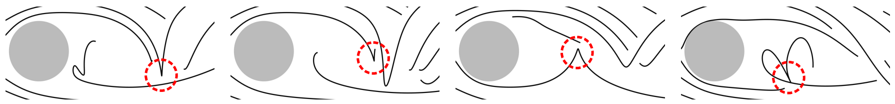


Fig. 6 Path lines in the flow behind a cylinder have been seeded using Algorithm 1, but without excluding the areas around cusp seeding lines. While the result is free of (self-)intersections, some path lines exhibit cusps or cusp-like shapes (highlighted by red circles).

Furthermore, we use a simple voxel bit mask to exclude certain space-time locations from the seeding of new field lines. This includes obstacles in the flow (such as a cylinder), and most importantly areas around cusp seeding lines: cusps communicate discontinuities in the flow which are actually not there, since cusps are due to the spatial projection and not due to the underlying flow. We exclude these areas as follows (see also Section 3):

- Track all critical points, which yields lines in space-time.
- Integrate two surfaces (forward/backward) from each of these lines.
- Render the surfaces into the voxel bit mask with a certain width such that larger areas around the actual cusp seeding lines are avoided.

Applying this algorithm yields cusp-free and intersection-free visualizations, but it turns out that turbulent parts of the domain are only sparsely or not at all covered since the field lines have numerous (self-)intersections there. We think that an expressive visualization of unsteady flows demands the inclusion of a small number of (self-)intersecting lines as a trade-off between visual clarity and a faithful representation of the flow. Hence, we let the user select a desired number of such lines from an automatically computed set optimized by length and domain coverage, and feed this information into the above algorithm such that the rest of the domain can be filled with non-intersecting field lines. In all visualizations of the following results section we render the selected self-intersecting lines with color (blue for path lines, red for streak lines) and halos to enhance their perception. Furthermore, we encode time into the width of the path lines, and τ into the width of the streak lines.

Streak lines are always seeded at a certain time step, but we allow the user to choose either a time step or a time interval for seeding path lines. A small time interval is suggested as it allows for a better temporal coherence between the path lines. However, due to the boundary and the cutting of the path lines in Algorithm 1, not all path lines will start or end in the same time steps.

Discussion of possible alternatives. Topological information might not be readily available in every visualization system. Simply applying Algorithm 1 without excluding the areas around cusp seeding lines, however, does not yield satisfying results as exemplified in Figure 6: cusps and cusp-like shapes are clearly visible. Nevertheless, an alternative to our topology-based approach can be formulated based on our theoretical findings from Sections 3 and 4: we know that cusps appear next to an area of self-intersecting path lines. To exclude them from the seeding, one could determine for every grid point whether it gives rise to a self-intersecting path line. If so, this grid point and a certain number of grid points in its neighborhood are excluded from the seeding. This would serve as the voxel bit mask described earlier, but without actually using topological information. We tested this and it gave results similar to the ones shown in the next section. However, this brute-force method needs more computation time (depending on the resolution of the voxel bit mask) than the topological approach.

6 Results

The following results have been computed single-threaded on a laptop with an Intel Core 2 Duo T9550 (2.66GHz). The timings for the topological analysis and the subsequent field line placement are given in the respective figure captions. All integrations have been done with a 4th order Runge-Kutta scheme and adaptive step size control.

Figure 7 shows the flow above a heated cylinder which has been simulated using The Gerris Flow Solver [7]. It is given on a $41 \times 70 \times 241$ uniform grid. The comparison between the particle animation and our path line placement clearly elucidates the differences between showing transient impressions and a static visualization comprising all time steps. Furthermore, the seeding of the particles turned out to be a cumbersome, time-consuming process since seeding location and density had to be properly adjusted to achieve the shown effect. With the new path line placement method we were able to produce a meaningful result within seconds.

Path and streak lines for a flow behind a cylinder [7] are shown in Figure 8 ($400 \times 50 \times 1001$ uniform grid). As reasoned earlier, one finds self-intersecting field lines near critical points of the underlying tangent curve vector field. The topology

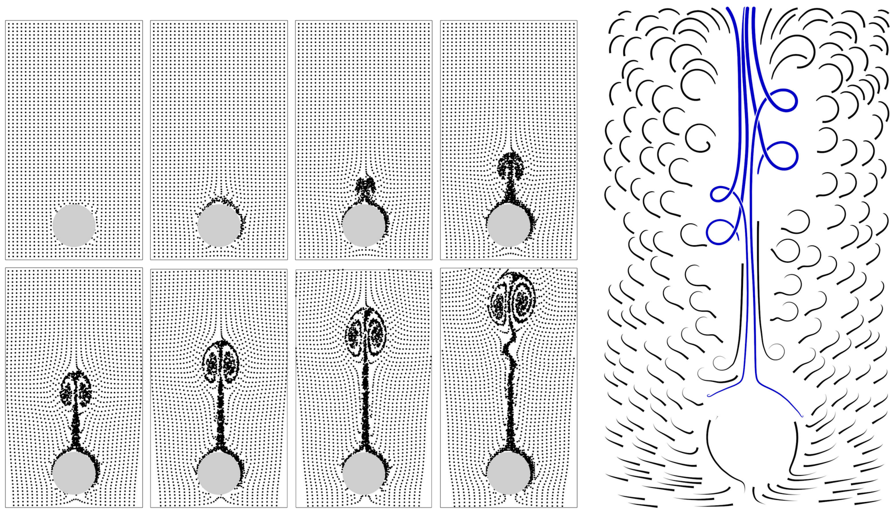


Fig. 7 Heated cylinder data set. Shown is a particle animation (left) and the result of our path line placement method (right). *Topology/Placement: 6/2 sec.*

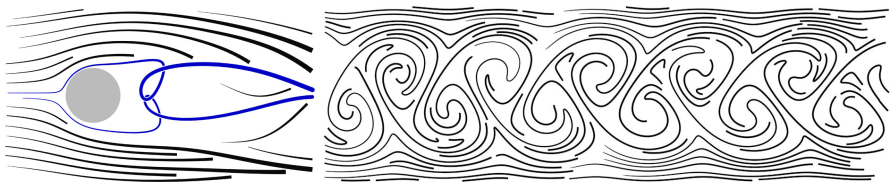


Fig. 8 Flow behind a cylinder. The path lines (left) are shown in the vicinity of the cylinder where critical points appear. *Topology/Placement: 2/1 sec.* The streak lines (right) are shown further downstream where the von Kármán vortex street is fully developed. Our algorithm was able to fill the domain with streak lines properly, i.e., no additional streak lines have been seeded. *Topology/Placement: 7/6 sec.*

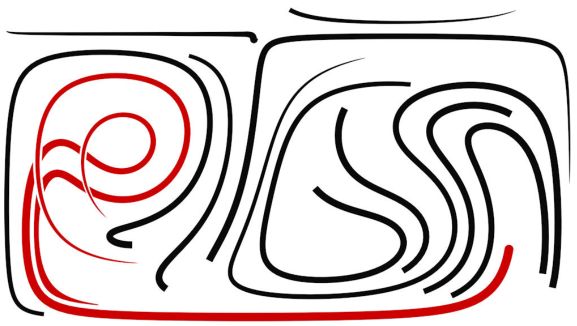


Fig. 9 Streak lines of the Double Gyre flow for $t = 10$ in the τ-interval $[-10, 0]$. The two red curves have been selected by the user from a set of pre-computed self-intersecting streak lines in order to fill the domain. Otherwise the left part would not have been filled by Algorithm 1 since it contains mainly self-intersecting streak lines. *Topology/Placement: 0.5/0.5 sec.*

of the original flow field has only a few critical points directly behind the cylinder, which makes the path lines interesting there while they become rather simple further downstream. The situation is different for streak lines: the spatial components of $\bar{\bar{\mathbf{q}}}$ have critical points where $\mathbf{v}(\phi_t^\tau(\mathbf{x})) = \mathbf{0}$. In other words, the critical points of the original flow have been "transported" downstream and create interesting streak line patterns there.

Figure 9 shows the streak lines of the well-known *Double Gyre* flow. We have chosen a rather long τ-interval for computing the streak line vector field ($256 \times 128 \times 1280$ uniform grid), which yields long streak lines in the final visualization. Two streak lines in the left part had to be selected by the user since this part contains only self-intersecting streak lines.

7 Conclusions and Future Work

We analyzed the spatial projections of characteristic curves in terms of cusps and self-intersections. These insights allowed us to develop a novel placement method for streak and path lines.

Future work needs to address the issue of intersecting field lines. Here, we avoided them as much as possible, but this may also leave certain areas less covered. Also, such visualizations could communicate other properties of the flow (curvature, vorticity, converging/diverging, etc.) by allowing a varying density of the shown field lines. Of course, incorporating intersections and dense areas into an image calls for new rendering approaches for these kinds of visualizations to maintain a certain visual clarity.

References

1. Cuntz, N., Pritzkau, A., Kolb, A.: Time-Adaptive Lines for the Interactive Visualization of Unsteady Flow Data Sets. Computer Graphics Forum **28**(8), 2165–2175 (2009)
2. Garth, C., Li, G.S., Tricoche, X., Hansen, C.D., Hagen, H.: Visualization of coherent structures in transient 2d flows. In: H.C. Hege, K. Polthier, G. Scheuermann (eds.) Topology-Based Methods in Visualization II, Mathematics and Visualization, pp. 1–13. Springer (2009). Topo-In-Vis 2007, Grimma, Germany, March 4 - 6
3. Jobard, B., Lefer, W.: Creating evenly-spaced streamlines of arbitrary density. In: Proceedings 8th Eurographics Workshop on Visualization in Scientific Computing, pp. 57–66. Boulogne (1997)
4. Kenwright, D.N., Lane, D.A.: Interactive time-dependent particle tracing using tetrahedral decomposition. IEEE Transactions on Visualization and Computer Graphics **2**(2), 120–129 (1996)
5. Marchesin, S., Chen, C.K., Ho, C., Ma, K.L.: View-dependent streamlines for 3d vector fields. IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization 2010) **16**(6), 1578–1586 (2010)
6. Mebarki, A., Alliez, P., Devillers, O.: Farthest point seeding for efficient placement of streamlines. In: Proc. IEEE Visualization 2005, pp. 479–486 (2005)
7. Popinet, S.: Free computational fluid dynamics. ClusterWorld **2**(6) (2004). URL http://gfs.sf.net/
8. Rosanwo, O., Petz, C., Prohaska, S., Hotz, I., Hege, H.C.: Dual streamline seeding. In: Proc. IEEE Pacific Visualization, pp. 9–16 (2009)
9. Sanna, A., Montrucchio, B., Arinaz, R.: Visualizing unsteady flows by adaptive streaklines. In: Proc. WSCG'2000. Plzen, Czech Republic (2000)
10. Theisel, H., Seidel, H.P.: Feature flow fields. In: Data Visualization 2003. Proc. VisSym 03, pp. 141–148 (2003)
11. Theisel, H., Weinkauf, T., Hege, H.C., Seidel, H.P.: Topological methods for 2D time-dependent vector fields based on stream lines and path lines. IEEE Transactions on Visualization and Computer Graphics **11**(4), 383–394 (2005)
12. Tricoche, X., Wischgoll, T., Scheuermann, G., Hagen, H.: Topology tracking for the visualization of time-dependent two-dimensional flows. Computers & Graphics **26**, 249–257 (2002)
13. Turk, G., Banks, D.: Image-guided streamline placement. In: Proc. Siggraph '96, pp. 453–460 (1996)
14. Weinkauf, T., Theisel, H.: Streak lines as tangent curves of a derived vector field. IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization 2010) **16**(6), 1225–1234 (2010). URL http://tinoweinkauf.net/. Received the Vis 2010 Best Paper Award
15. Weiskopf, D.: Dye Advection Without the Blur: A Level-Set Approach for Texture-Based Visualization of Unsteady Flow. Computer Graphics Forum (Eurographics 2004) **23**(3), 479–488 (2004)
16. Wiebel, A., Scheuermann, G.: Eyelet particle tracing - steady visualization of unsteady flow. In: Proc. IEEE Visualization 2005, pp. 607–614 (2005)
17. van Wijk, J.J.: Image based flow visualization. In: Proc. ACM SIGGRAPH '02, pp. 745–754 (2002)