Streamline-based topological graph construction with application to self-animated images

RENATA NASCIMENTO AND THOMAS LEWINER

Department of Matematics — Pontifícia Universidade Católica — Rio de Janeiro — Brazil
zeus.mat.puc--rio.br/{renata,tomlew}.

Abstract. Vector field analysis and visualization is a fundamental tool in science and engineering applications, raising the need for robust methods to capture and represent the global vector field behavior. One such representation, the topological graph, partitions the domain into regions where the flow of the vector field is homogeneous. This work introduces a topological graph construction based on streamlines. It is guaranteed to produce a coherent result even when some singularities are not detected. This work also details an application of topological graphs to improve the generation of self-animated images. In this application, the streamline-based approach carries almost no overhead, since self-animated images already rely on streamlines, but leads to a threefold speed-up of the core processing.

Keywords: ***Vector Field. Vector Field Topology. Visualization. Streamline. Topological Graph. Segmentation. Topological Method. Self-Animated Image.***

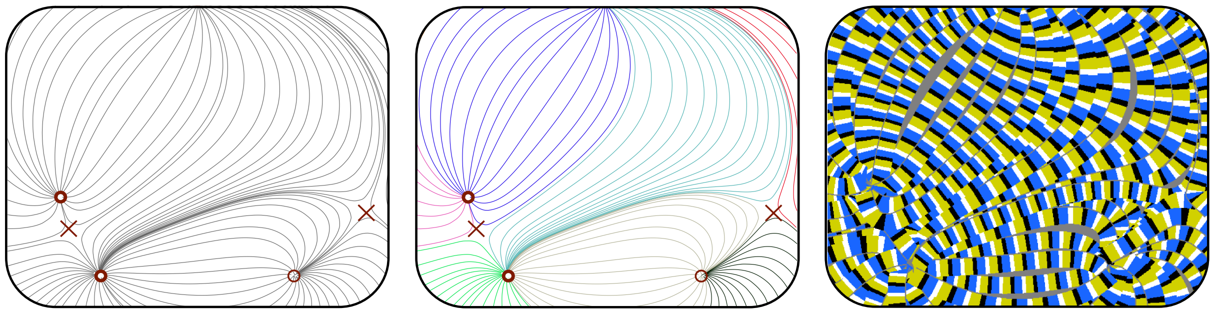


Figure 1: *The topological graph of a vector field is built from a set of its streamlines (left) by grouping streamlines with similar behavior. This partition captures the global behavior of the vector field (middle). This information can then be used directly to improve visualization techniques using streamlines, such as optimizing self-animated images generation (right).*

1 Introduction

Vector field analysis and visualization became a fundamental tool in science and engineering, raising interest not only from the visualization community [6], but also from many applications as fluid mechanics [17] or medicine [15]. While understanding the global dynamics of a complex vector field remains a challenge, in particular for turbulent flows, there has been significant improvements on the visualization of vector fields, for example using glyphs [5], streamlines [20, 11, 13, 10, 14], or illustrative images [3, 21, 4].

Most of those advances heavily rely on a better understanding and more robust computing of the vector field topology [18, 16]. This topology splits into the vector field's singularities and the behavior in-between them, which is essentially tubular, i.e. locally equivalent to a constant field [8]. Thus, a simple way to capture the field behavior consists in partitioning the domain into regions of tubular flow (Figure 1). This segmentation is called the topological graph of the vector field [19]. This work introduces a construction of the topological graph on top of a set of streamlines. Depending on the density of streamlines used, our construction may not be exact, but it is guaranteed to be *coherent*, i.e. it builds the topological graph of some simplified version of the field. This streamline-based method can be benefited from the advances in streamline generation [10, 14].

Moreover the topological information extracted with this strategy is directly useable to streamline-based applications. This work explores one example of such use, integrating the streamline-based topological graph construction to accelerate the generation of self-animated image [4]. This visualization technique uses optical illusions to create a movement effect that follows a vector field (Figure 1(right)). However, it relies on a slow brute force optimization. For this application, the use of the topological graph speeds up the original optimization by a factor above 3, almost without overhead since the self-animated image already relies on streamlines.

Preprint MAT. 15/13, communicated on April 29^{th}, 2013 to the Department of Mathematics, Pontifícia Universidade Católica — Rio de Janeiro, Brazil. The corresponding work was published in the proceedings of Sibgrapi 2013. IEEE Press, 2013.

2 Related work

In this section we review some work related to streamline generation, vector field topology visualization and self-image images. We refer to the book of Hansen and Johnson [6] for a greater review of vector field visualization.

Streamlines placement The global behavior of a vector field is described by its flow and the associated streamlines. The efficiency of vector field analysis and visualization through streamlines highly depends on their density and placement: too many streamlines generate clutter and redundant analysis, while too few may lead to a incomplete representation. Turk and Banks [20] pioneered this area by progressively placing streamlines to evenly distribute them. This method was later improved using a single pass [11], to favor long [13] or more uniform [10] streamlines. The present work can use any streamline placement method and can benefits from further improvements in that area.

Topological methods Another way to analyze the vector field relies on topological methods to classify and localize the different flow behaviors, in particular at the singular points [7]. The dynamic in-between singular points can be segmented into regions of tubular flow. Tricoche [19] proposed to represent this segmentation through the topological graph, computed from a few specific streamlines. Such methods can be extended to produce simplified representations that capture the more persistent behaviors [18]. The topological information can also be used to assist the user for understanding [5] or smoothing [16] the vector field. The computations of topological information, in particular the topological graph may suffer from numerical imprecision [19], motivating error-controlled streamline integrators [14]. This work proposes a topological graph construction based on streamlines that always generate a coherent result, even if the streamline set is too sparse to capture all the singularities.

Self-animated images Wei [21] proposed the use of optical illusion to visualize vector fields. The perceptual motion is caused by an asymmetric pattern of repeated colors. Chi *et al.* [4] extended this approach by using the pattern along the streamlines, optimizing the pattern placement. As an application of this work, the streamline-based topological graph is exploited to speed up the generation of self-animated images, accelerating its core optimization by a factor above 3.

3 Continuous vector field topology

This section reviews the basic concepts of vector field topology. We refer the reader to the books of Andronov [1] and Hirsch and Smale [8] for a complete discussion, and to the work of Tricoche [19] for the definition of topological graph.

(a) Vector field, flow and streamline

Definition 1 (Vector field) *A* vector field $\mathbf{v}$ *in a planar domain* $D \in \mathbb{R}^2$ *is a function that maps each point* $\boldsymbol{x} = (x, y) \in D$ *to a bi-dimensional vector* $\mathbf{v}(\boldsymbol{x}) = (v^x(x, y), v^y(x, y)) \in \mathbb{R}^2$.

A vector field is usually understood as a steady velocity field, where particles are advected by the *flow* [1].

Definition 2 (Flow) *The* flow $\phi : U \subset \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^2$ *associated to a vector field* $\mathbf{v}$ *is defined for all time* $t \in \mathbb{R}$ *by the following differential equation.*

$$\frac{\partial}{\partial t}\phi(\boldsymbol{x}, t)\,|_{t=\tau} \;=\; \mathbf{v}(\phi(\boldsymbol{x}, \tau)) .$$

The flow maps the position $\mathbf{x}$ of a particle to its position after being advected during a given period of time t. Varying time t, we obtain the trajectory of the particle, referred as the *streamline* passing through $\mathbf{x}$.

Definition 3 (Streamline) *The* streamline $sl(\boldsymbol{x})$, *passing through* $\boldsymbol{x}$ *with* $\mathbf{v}(\boldsymbol{x}) \neq 0$, *is defined as:*

$$sl(\boldsymbol{x}) = \bigcup_{t=-\infty}^{\infty} \phi(\boldsymbol{x}, t) \quad .$$

If the vector field vanishes at $\boldsymbol{x}_0$, *i.e.* $\frac{\partial}{\partial t}\phi(\boldsymbol{x}_0, t) = \mathbf{v}(\boldsymbol{x}_0) = \boldsymbol{0}$, *then* $sl(\boldsymbol{x}_0) = \{\boldsymbol{x}_0\}$.

(b) Singular point

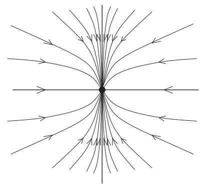
Definition 4 (Singular point) *A point* $\boldsymbol{x}_0 \in D$ *is called* singular *for* $\mathbf{v}$ *if* $\mathbf{v}$ *vanishes at* $\boldsymbol{x}$, *i.e.* $\mathbf{v}(\boldsymbol{x}_0) = (0, 0)$.

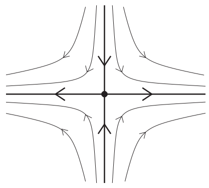
A streamline can converge when $t \to \pm\infty$, in which case its endpoint is a singular point. Since streamlines are integral lines of the flow differential equation (Def. 2), they do not intersect except at singular points [1].

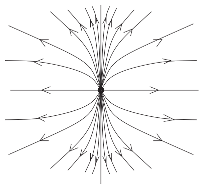
Assuming that the vector field $\mathbf{v}$ is differentiable, Hartman-Grobman theorem [8] partially classifies the local behavior around a singular point $\mathbf{x}_0$ from the eigenvalues of the Jacobian matrix of $\mathbf{v}$ at $\mathbf{x}_0$:

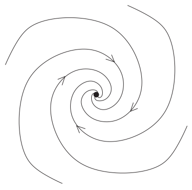
$$J_{\mathbf{v}}(\mathbf{x}_0) = \begin{bmatrix} \frac{\partial}{\partial x} v^x(\mathbf{x}_0) & \frac{\partial}{\partial y}\, v^x(\mathbf{x}_0) \\ \frac{\partial}{\partial x} v^y(\mathbf{x}_0) & \frac{\partial}{\partial y}\, v^y(\mathbf{x}_0) \end{bmatrix} \quad .$$

Throughout this work, we will consider that the vector field is well approximated by its Jacobian at the singular points, i.e. the real parts of its eigenvalues does not vanish. We will also not consider closed streamlines. For the bi-dimensional case, this linear classification of singular points is illustrated in Figure 2.


(a) *$R_1, R_2 < 0$ and $I_1 = I_2 = 0$: sink.*


(b) *$R_1 > 0, R_2 < 0$ and $I_1 = I_2 = 0$: saddle.*


(c) *$R_1, R_2 > 0$ and $I_1 = I_2 = 0$: source.*


(d) *$R_1, R_2 < 0$ and $I_1, I_2 \neq 0$: spiral sink.*

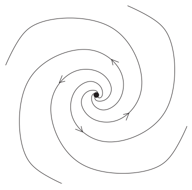

(e) *$R_1, R_2 > 0$ and $I_1, I_2 \neq 0$: spiral source.*

Figure 2: *Singularities classification following Hartman-Grobman theorem, where $(R_1 + \mathbf{i}\, I_1)$ and $(R_2 + \mathbf{i}\, I_2)$ are the complex eigenvalues of $J_{\mathbf{v}}(\boldsymbol{x}_0)$.*

(c) Separatrix, topological graph

In-between singular points, the flow is essentially tubular [8], where adjacent streamlines generally start from the same source and end at the same sink. This constant behavior changes at the separatrices, where the source or the sink suddenly changes for a saddle.

Definition 5 (Separatrix) *A* separatrix *is a streamline that starts or end at a saddle $\boldsymbol{x}_0$. At the singular point $\boldsymbol{x}_0$, the separatrix is tangent to an eigenvector of $J_{\mathbf{v}}(\boldsymbol{x}_0)$.*

When the vector field is defined in the whole plane, its global behavior can be summarized in its *topological graph*.

Definition 6 (Topological graph) *The* topological graph *of a vector field $\mathbf{v}$ is a graph whose nodes are the singular points of $\mathbf{v}$ and whose arcs are its separatrices.*

In particular, without closed streamline or non-linear singularities, the topological graph arcs bound regions with two saddles, a source and a sink. We will refer to this property as the *coherence* of the topological graph.

The definition of a topological graph for bounded domains is slightly more delicate, since some streamlines are cut by the boundary. In this case, Tricoche [19] proposed to add extra nodes to the topological graph for the endpoints of the streamlines cut by the boundary. If two adjacent streamlines cut the boundary in the same manner (pointing in- or outwards, or being tangent), their nodes are merged, as illustrated in Figure 3. The merged extra nodes then correspond to segments of the boundary of D, and are called *boundary source* when the adjacent streamlines point out of D, *boundary sink* when they point to the inside of D and *boundary*

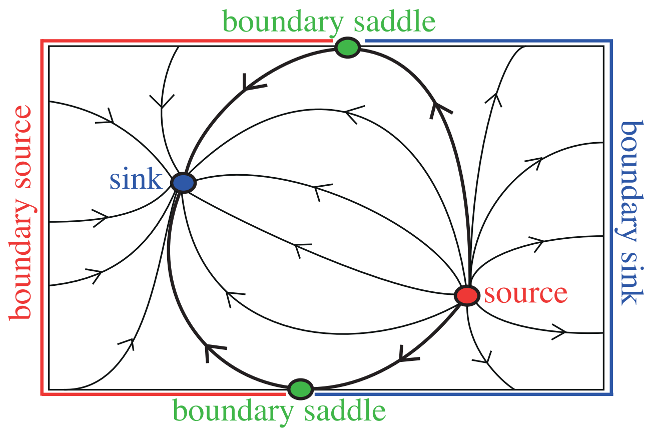


Figure 3: *Topological graph of a bounded domain D: the boundary of D is split into boundary sources, sinks and saddles.*

saddle when a streamline is tangent to the boundary of D. This construction respects the definition of coherence in the case of bounded domain.

For a bounded domain, the topological graph also induces a decomposition of the domain into regions of equivalent streamlines, bounded by separatrices and the domain boundary. Throughout this work we will refer to the topological graph as this domain decomposition, since it contains more information on the boundary of the domain than the graph.

4 Topological graph construction from streamlines clustering

Tricoche [19] proposed to construct the topological graph by first computing all the singular points. The separatrices are obtained by integrating the flow equation starting tangent to every saddle $\mathbf{x}_0$ in the direction of the eigenvectors of $J_{\mathbf{v}}(\mathbf{x}_0)$, as in Def. 5.

Although those elements are topologically stable, they may be hard to compute numerically. In particular, detecting all singularities of an analytic field may require infinite precision. If a saddle is not detected, and its separatrices missing, then the topological graph may become incoherent.

In this work, we propose to build on the existing work on streamlines generation to construct the topological graph. The streamlines are clustered in order to obtain the same decomposition of the domain as the topological graph. On the one hand, the result will depend on the streamline generation method, in particular on the generated streamline density. On the other hand, the topological graph is always coherent, even if some singular points are not detected.

The streamline clustering is straightforward for unbounded domains. For bounded domains, we propose to progressively partition the boundary, increasing the complexity of the captured interactions between boundary singularities, as follows.

Step 0: Streamline extensions. We first generate a set of streamlines. Most vector field visualization methods try to capture the complexity of a given vector field, so we can use any streamline placement and generation technique. In particular, they tend to approximate the separatrices, which helps our generation of the topological graph, although it is not necessary. Usually, those methods do not fully integrate the streamlines to avoid clutter, but here we extend those streamlines until we reach a singular point, the boundary, or eventually loop. Although we do not display those extensions, we use it to assign to each streamline its start and end, those being singular point or boundary.

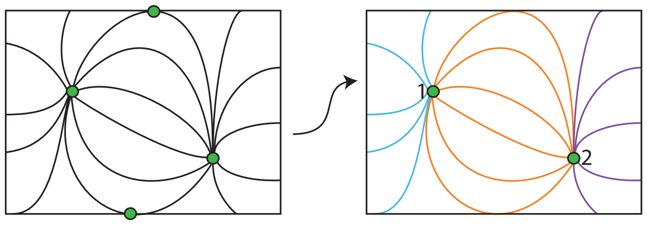


Figure 4: *Step 1: segmentation of the streamlines considering the boundary as a single singularity. The streamlines are segmented into three groups: orange, purple, and blue.*

Step 1: Boundary as a single singularity. We first create a single node for the boundary and a node for every source and sink. We group the streamlines with the same starting and ending node. For simple examples, this already gives the correct domain partition, as in Figure 3, with three groups of similar streamlines: source 2 to sink 1, source 2 to the boundary and boundary to sink 1.

Step 2: Boundary subdivision into connected components. Among the streamlines crossing the boundary, the previous step distinguishes those originating or ending at different singularities (Figure 6). However, the regions delimited by the topological graph are simply connected. Therefore, the intersection of a group of streamlines with the boundary should be a (connected) segment. To ensure this property, each group is subdivided along the connected components of the boundary. For example in Figure 5, the streamlines of Group 1 are split into Groups 1 and 4 depending on whether they cross the boundary through the top and sides of the domain (Group 1) or through the bottom of the domain (Group 4). This step is efficiently performed by first storing at the boundary pixels which streamlines are crossing, together with its other end-node (that may be a boundary segment). Then we walk along the boundary and split the group of streamlines each time the other end-node changes.

Step 3: Boundary to boundary streamlines. After Step 2, adjacent streamlines crossing the boundary at their beginning and end belong to the same group. However, a separatrix may exist between them, so that they should belong to different groups (Figure 6). We prefer to avoid marking the streamlines crossing the boundary as entering or going out of the domain, which is subject to numerical instabilities near boundary saddles. Therefore, we walk again (or simultaneously with Step 2) along the boundary, queue each new streamline we cross and pop it the second time it is crossed.

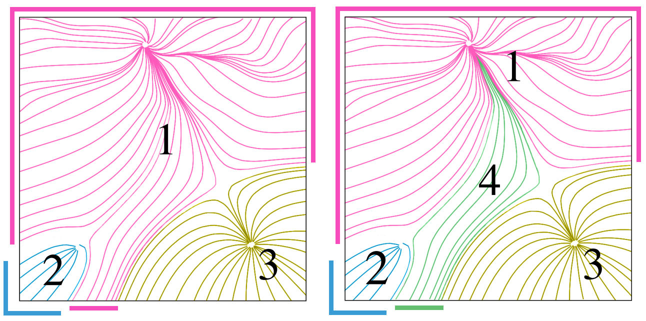


Figure 5: *Step 2: the streamlines of Group 1 (left) all start at the same singularity and all cross the boundary. However, the intersection of Group 1 and the boundary is disconnected, and new groups (here Groups 1 and 4) are created for each connected component (right).*

Since the streamline group we would like to refine cross the boundary twice, and since those streamlines do not intersect inside the boundary, the queue will necessarily pop a whole group before entering the next one. We thus create a new group each time the queue size passes a minimum. As a comparison, Figure 6 shows our construction and the topological graph computed from the separatrices, following Tricoche's approach [18].

Guarantees. A generated streamline may cross the boundary zero, one or two times. Step 1 guarantees at least the classification for streamlines that do not cross the boundary. Step 2 guarantees the classification of streamlines crossing the boundary once, and may correctly classify the boundary-to-boundary streamlines. Step 3 guarantees the classification of the latter, grouping streamlines in a coherent manner. Finally, if all the singularities are detected and if enough streamlines are generated, the groups of streamlines correspond to the decomposition of the domain induced by the topological graph.

5 A simple implementation

In order to show the feasibility of the topological graph computation proposed in the previous section, we describe here a very simple implementation for vector fields sampled on regular bi-dimensional grids with bilinear interpolation and equally-spaced streamlines [11].

In that case, vector values associated to points (x_i, y_j) of a regular grid is denoted $\mathbf{v}_{ij} = (v_{ij}^x, v_{ij}^y) = \mathbf{v}(x_i, y_j)$. The bilinear interpolation $\mathbf{b}_{ij}$ returns the value inside a grid cell. Up to a translation of vector $(-x_i, -y_j)$ and a scaling of $\frac{1}{x_{i+1}-x_i}$ along the x axis and of $\frac{1}{y_{i+1}-y_i}$ along the y axis, we can assume that coordinates x and y both vary from 0 to 1 inside grid cell $(i, j) = (0, 0)$, which simplifies the expression for $\mathbf{b}$:

$$\begin{aligned} \mathbf{b}_{00} &: [0,1]^2 \to \mathbb{R}^2, \\ \mathbf{b}_{00}(x,y) &= (1-x)(1-y)\cdot\mathbf{v}_{00} + x(1-y)\cdot\mathbf{v}_{10} \\ &\quad +(1-x)\;y\;\cdot\mathbf{v}_{01} + x\;y\;\cdot\mathbf{v}_{11}\ . \end{aligned}$$

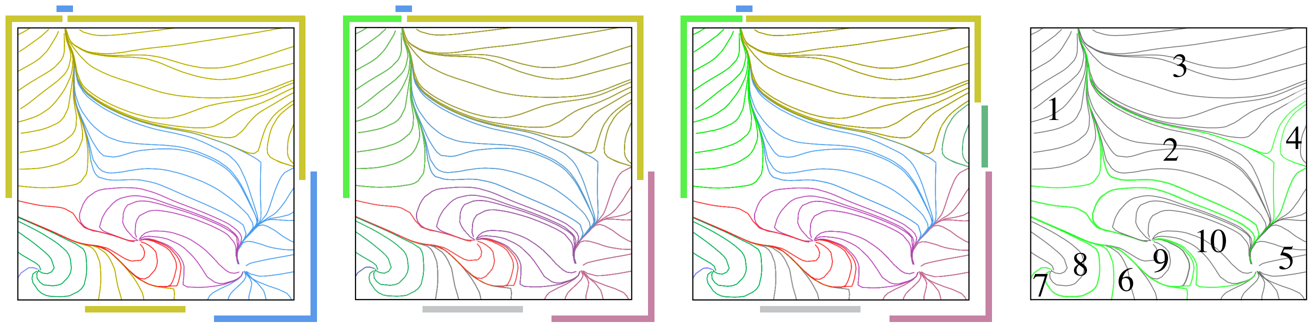


Figure 6: *Three steps of the topological graph refinement (left), matching the one generated from the separatrices [18] (right). Observe that at Step 3 (middle left to middle right), adjacent boundary to boundary streamlines may be separated by a separatrix (here at the right side of the domain).*

The singularities can be detected based on the bilinear interpolation based on a direct solution of $\mathbf{v}(x, y) = \mathbf{0}$, the winding number or noise-tolerant methods [16].

For example, the first option is equivalent to solving system $\mathbf{b}_{00}(x, y) = (0, 0)$, which is equivalent to finding the roots of the polynomial in y:

$$\begin{aligned} (& -v^x_{01}\, v^y_{00} + v^x_{01}\, v^y_{10} + v^x_{11}\, v^y_{00} - v^x_{11}\, v^y_{10} + \\ & + v^x_{00}\, v^y_{01} - v^x_{00}\, v^y_{11} - v^x_{10}\, v^y_{01} + v^x_{10}\, v^y_{11} \quad) \cdot y^2 \\ + (& \; 2\, v^x_{01}\, v^y_{00} - 2\, v^x_{00}\, v^y_{01} - v^x_{11}\, v^y_{00} - \\ & - v^x_{01}\, v^y_{10} + v^x_{10}\, v^y_{01} + v^x_{00}\, v^y_{11} \quad) \cdot y \\ + & \quad v^x_{00}\, v^y_{01} - v^x_{01}\, v^y_{00} \quad . \end{aligned}$$

For each root y, the associated coordinate x value is:

$$x = \frac{(v^y_{00} - v^y_{01}) \cdot y - v^y_{00}}{(v^y_{00} - v^y_{10} - v^y_{01} + v^y_{11}) \cdot y - v^y_{00} + v^y_{10}} \, .$$

If both the root y and the associated x are inside $[0, 1]$, there is a singular point in the cell at (x, y). The singularity type is computed from the eigenvalues of the Jacobian according to Sec. 3(b).

We use one of the first and most simple streamline generation algorithm due to Jobard and Lefer [11], which produces evenly-spaced streamlines. Starting from a random seed, a streamline is integrated, leaving seeds on both sides at a fixed distance d and filling an occupation grid of width $\frac{d}{2}$. From those seeds, streamlines are integrated in the same fashion, but the integration stops when the streamline enters the occupied cells of the grid. We use a Runge-Kutta scheme of fourth-order with the bilinear interpolation for the streamline integration.

In order to capture the topological graph even with a sparse set of streamlines (i.e. a large value of d), we put seeds near the saddles, in the direction given by the eigenvectors of the Jacobian matrix at the saddle point. This is not enough to guarantee the generation of all separatrices as discussed earlier, but even without all of them, the topological graph will be coherent.

6 Application to self animated images

In this section we propose a direct application of our streamline-based topological graph construction to improve the generation of self-animated images. Although the original work used a refined streamline placement [13], we generate all the results using the method described in the previous section, in order to maintain this paper self-contained.

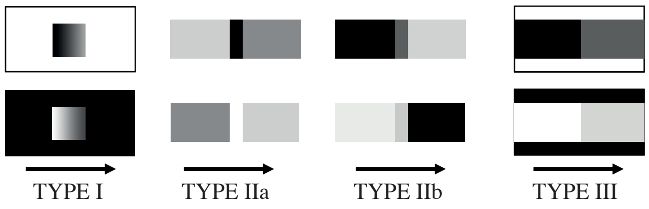


Figure 7: *Types of Fraser-Wilcox illusion (extracted from [4]).*

(a) Self-animated images from streamlines

Chi *et al.* [4] generated self-animated images using optimized Fraser-Wilcox optical illusions [12]. Such illusions rely on Repeated Asymmetric Patterns *RAP* [2], which causes a motion impression to most observers. Chi *et al.* apply the type IIa pattern (Figure 7) to render the streamlines, which creates an illusory motion following the vector field.

To do so, each streamline is divided into segments of equal length *SegLen*. The segments are colored according to a Type II optimized Fraser-Wilcox illusion respecting the 1:2:1:2 rule. In our results, we use such colored RAP: $\{$*Black-Blue-Blue-White-Yellow-Yellow*$\}$. However, the illusory motion effect of the RAP is affected by the patterns on close-by streamlines. Bad pattern coordination produces color blocks disturbing the illusion (Figure 8(top)). The pattern assignment, in particular the initial color and the segment length *SegLen*, must be optimized to avoid such blocks. We propose an alternative to the optimization proposed by Chi *et al.* [4] using our streamline-based topological graph.

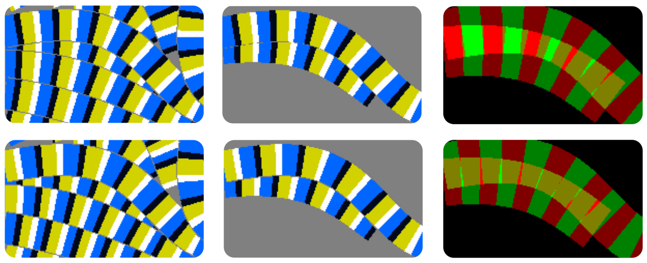


Figure 8: *Badly coordinated pattern (top line) gather color blocks across streamlines. Pattern placement is optimized one streamline at a time (middle). The quality of the pattern placement is measured rendering a red/green pattern instead of dark/light, and counting overlapping yellow pixels (illustration extracted from [4]).*

(b) Original optimization

Chi *et al.* [4] optimized the pattern placement by maximizing the difference of intensities between RAP fragments of close-by streamlines. To compute this quality measure, they render an optimization image blending the streamlines with their current RAP parameters, but using red for the darker parts of the RAP and green for the lighter ones (Figure 8). The quality of the RAP placement increases when more light parts of the RAP of one streamline overlap the dark parts of its adjacent streamlines, and vice-versa, i.e. when the optimization image of the streamlines has more yellow pixels (Figure 8).

There are basically two parameters in the optimization: the initial color of the segment and the length of each segment *SegLen*. To avoid large discrepancies, *SegLen* may vary by a factor between 0.8 and 1.3 of the initial length. The optimization starts with a random streamline, placing its RAP with a random initial color and the initial *SegLen*. Then, several RAP initial colors and *SegLen* assignments for the closest streamline are tested in a Monte Carlo fashion, keeping the one that maximized the quality so far. This image-based optimization is then repeated on an adjacent streamline until processing all the streamlines. Eventually, no RAP parameter for the current streamline can satisfy a minimal quality, leading to a deadlock (Figure 9(left)). In this case, the whole process starts over with a new streamline, randomly chosen. This rebooting of the optimization is quite frequent in practice, and greatly harms the performance of the optimization.

(c) Optimization using streamline-based topological graphs

Although Chi *et al.* [4] obtain high quality images, their method is computationally intensive. Indeed, the brute force optimization has a very slow convergence (Tab. 1). Some choices of the initial streamline surely lead to a deadlock (Figure 9(left)), which significantly increases the time used to generate the self-animated image. We propose here to extract from the topological graph an ordering of the deadlines that usually avoids deadlocks.

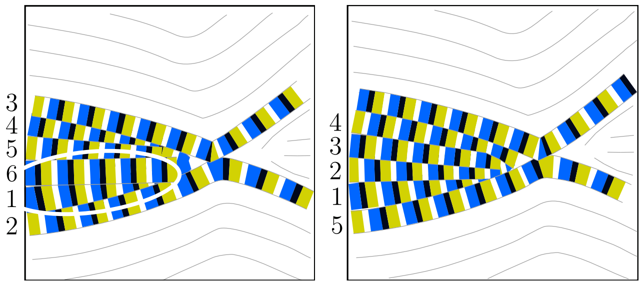


Figure 9: *Optimization performance depending on the initial streamline. (left) Following the order from 1 to 6 creates of a vertical pattern between the first and last one that cannot be avoided by changing the start color (from blue to yellow), or segment length of streamline 6. (right) A different order leads to a high contrast greedily.*

In the topological graph decomposition, all the streamlines of the same group have the same behavior, equivalent to a tubular flow [1]. Optimizing the RAP placement inside a group is thus straightforward and deadlock-free.

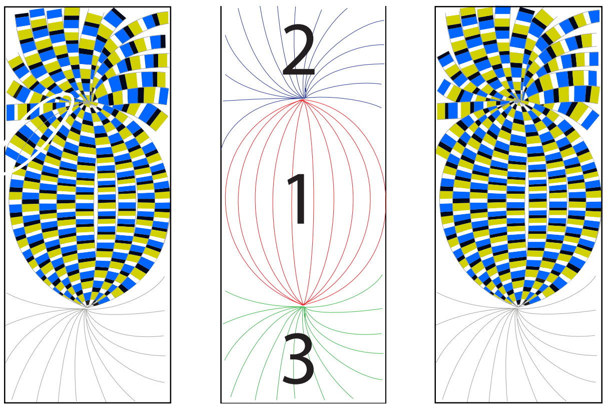


Figure 10: *Optimization through the topological graph: when a conflict between streamlines is detected (left), it can be solved by starting over only the RAP placement on the streamline group, here Group 2 (middle), leading to a high quality result without recomputing Group 1 (right).*

Deadlock may occur between groups, usually near singular points. With the topological graph, there is no need to systematically start the whole optimization from scratch. Indeed, only the RAP placement on the streamlines of one group (preferentially the smallest) needs to be re-optimized (Figure 10). We thus optimize the RAP placement group by group. Furthermore, in each group we optimize first the separatrices (or the closest streamlines to the transition), since those may create deadlocks if processed after the group.

We use the same image-based RAP optimization on a single streamline, but the streamline-based topological graph allows us to decouple the optimization per group, significantly accelerating the convergence of the algorithm as shown in the next results section.

Table 1: *Experimental results on a core 2 duo laptop: the topological graph computation has very little overhead, while its use significantly accelerates the self-animated image optimization.*

Data	Fig.	Number of samples	Number of singularities	Number of regions	Topological graph	Proposed optimization	Chi *et. al.* [4] optimization	Gain
2 singularities	Figure 13	2,500	2	5	2 ms	1.42s	3.07 s	2.2x
4 singularities	Figure 13	2,500	4	9	2 ms	1.30s	3.00 s	2.3x
6 singularities	Figure 11	2,500	6	9	3 ms	1.54s	3.77 s	2.4x
canceling singularities	Figure 12	2,500	5	4	4 ms	1.57s	3.98 s	2.5x
PIV 1	Figure 14	15,624	9	12	300 ms	9.26s	32.45 s	3.5x
PIV 2		15,624	28	15	300 ms	6.96s	38.92 s	5.6x

7 Results

We tested our topological graph construction mainly on analytical fields (Figs. 12, 13 and 11) where we could validate the correctness of the result, and on a field measured by a PIV method from fluid dynamics experiments (Figure 14). In all cases, the computation of the topological graph is extremely fast (Table 1). As expected, its speed decreases with the size of the image and the complexity of the vector field.

The correctness of the result depends on the streamline placement algorithm, in particular its density, as can be observed in Figure 12. However, at all the resolutions the topological graph is coherent, even when some singular points are not detected.

We also tested our optimization for self-animated images. To compare with the algorithm proposed by Chi *et al.* [4], we fix a target quality of the image, measured as the number of non-yellow pixels in the optimization image. In all the results presented here the target was 30% of yellow pixels. With this target quality criterion, the self-animated images obtained by both methods have very similar quality, as observed in Figure 13. Furthermore, the generated self-animated images capture most of the vector field dynamics even with more singularities (Figure 11).

We report the time spent by each optimization to reach the target quality in Table 1. The use of the streamline-based topological graph improves the execution time of the optimization by a factor greater than two on simple examples. Moreover, since self-animated images already require a set of streamlines, the use of the topological graph has a very little overhead. On data with more complex topology, the gain increases substantially. For example on the PIV example (Figure 14), the gain reaches a factor of 3.5.

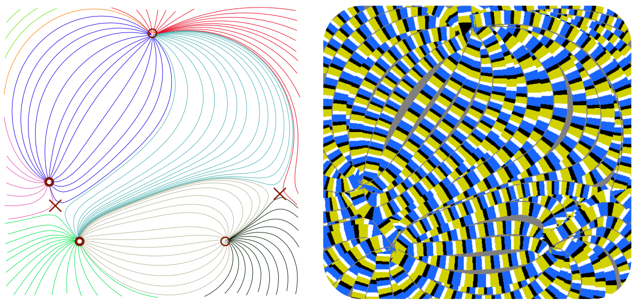


Figure 11: *Topological graph (left) and self-animated images (right) on a synthetic field with six singularities.*

8 Discussion

Aiming at assisting vector field analysis and visualization, this work proposes a topological segmentation built on top of existing streamline placement technique. This domain decomposition is equivalent to the topological graph, identifying streamlines with similar behaviors. This construction suits particularly for methods that already use streamlines, such as self-animated images generation [4], where it speeds up the image optimization time by several factors.

The current method still has some limitations and several possible extensions. First of all, we do not consider closed streamlines, which is a non-local type of singularities that would affect the topological graph. Such closed streamlines can be efficiently detected in our implementation context [9], and require a treatment in our segmentation similar to the boundary. We also aim to handle higher order (non-linear) singularities for the streamline seeding. Second, the method relies on a streamline generation. Although recent work provides some topological guarantees on the streamline integration [14], a low density of streamlines may still lead to a coherent but incorrect result (Figure 12). Finally, our current implementation handles vector fields sampled on regular grid, which is well suited for image. Most of our construction extends straightforwardly to our method to unstructured grid, mesh-less data or higher dimensional vector fields.

Acknowledgements. The authors would like to thank João Paixão for his contribution to this work, and to CNPq, FAPERJ and CAPES for partially financing this work.

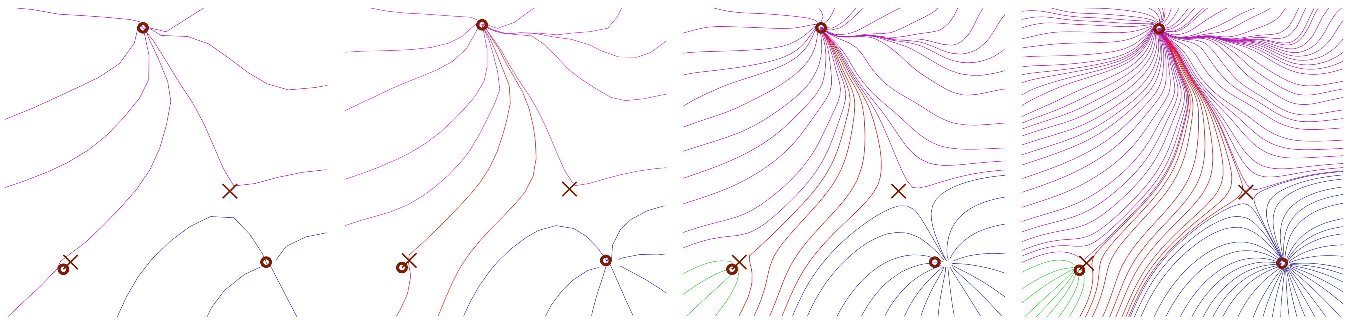


Figure 12: *Topological graph on a synthetic field with almost canceling singular points (left of each image), increasing the streamlines densities (from left to right): $d = 0.15, 0.2, 0.4, 0.9$. Although the graph is not exact for low densities, it always define coherent regions delimited by two saddles (indicated with a cross) , a source, and a sink (as a boundary segment, or indicated with a circle).*

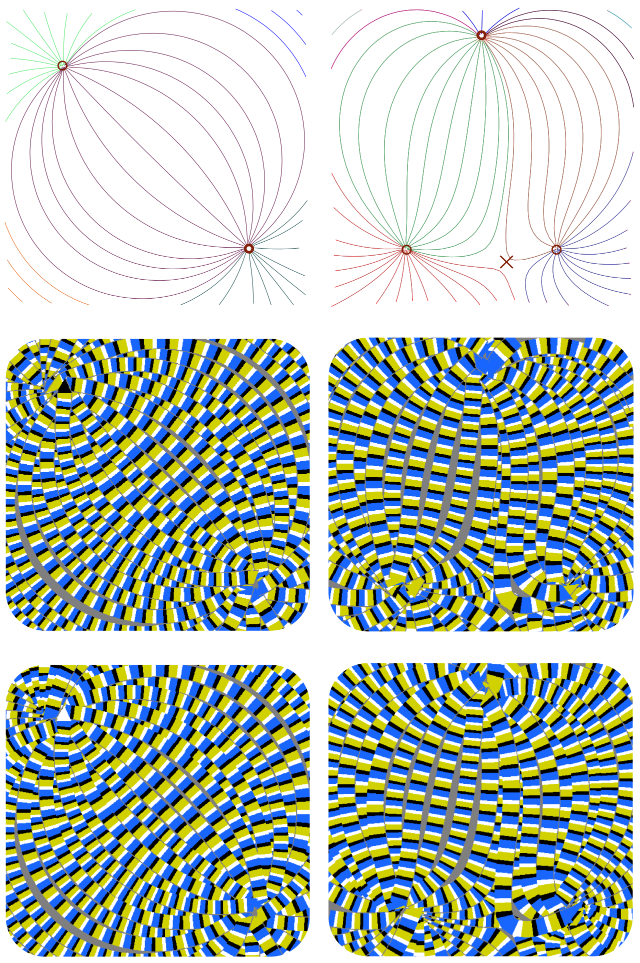


Figure 13: *Topological graph (top) and self-animated images generated by the original optimization (middle) and our optimization (bottom) on a synthetic field with two singularities (left) and four singularities (right).*

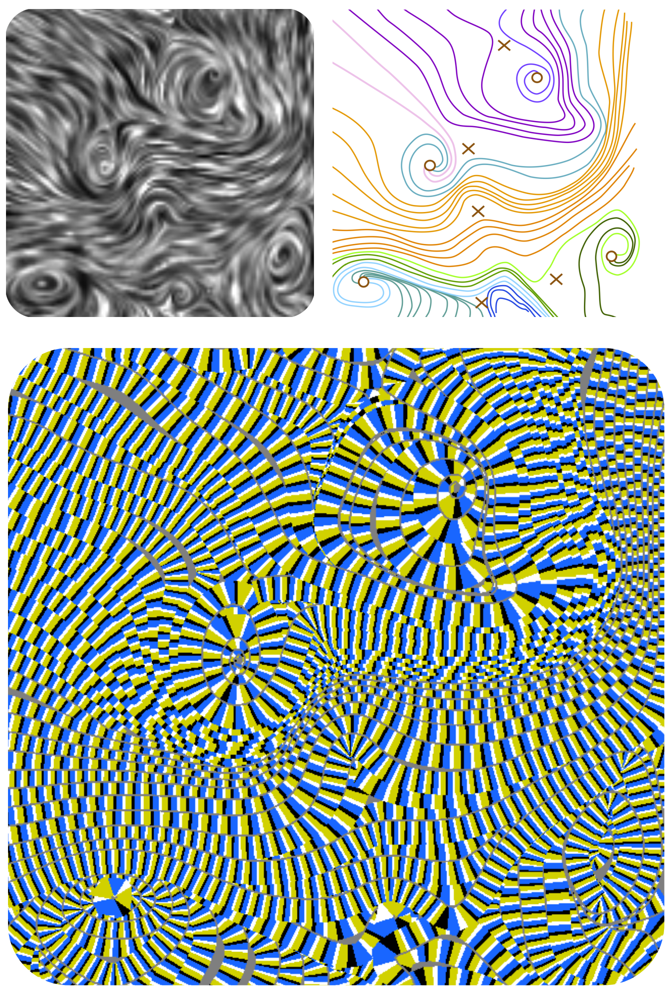


Figure 14: *LIC view [3] (top left), topological graph (top right) and self-animated images (bottom) on a smoothed PIV data.*

References

[1] A. Andronov. *Qualitative Theory of Second-Order Dynamic System*. Israel Program for Scientific Translations, 1973.

[2] B. Backus and I. Oruc. Illusory motion from change over time in the response to contrast and luminance. *Journal of Vision*, 5, 2005.

[3] B. Cabral and L. Leedom. Imaging vector fields using line integral convolution. In *Siggraph*, pages 263–270, 1993.

[4] M. Chi, T. Lee, Y. Qu and T. Wong. Self-animating images: Illusory motion using repeated asymmetric patterns. In *Siggraph*, page 62, 2008.

[5] J. Daniels II, E. Anderson, L. Nonato and C. Silva. Interactive vector field feature identification. *Transactions on Visualization and Computer Graphics*, 16(6):1560–1568, 2010.

[6] C. D. Hansen and C. R. Johnson. *The visualization handbook*. Academic Press, 2005.

[7] J. Helman and L. Hesselink. Visualizing vector field topology in fluid flows. *Computer Graphics and Applications*, pages 36–46, 1991.

[8] M. Hirsch and S. Smale. *Differential Equations, Dynamical Systems, and Linear Algebra*. Academic Press, 1974.

[9] H. Theisel, T. Weinkauf, H.-C. Hege and H.-P. Seidel. Grid-independent detection of closed stream lines in 2d vector fields. In *Vision, Modeling and Visualization*, pages 421–428, 2004.

[10] O. Rosanwo, C. Petz, S. Prohaska, H.-C. Hege and I. Hotz. Dual streamline seeding. In *PacificVis*, pages 9–16, 2009.

[11] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. In *Visualization in Scientific Computing*, pages 45–55, 1997.

[12] A. Kitaoka. Anomalous motion illusion and stereopsis. *Journal of Three Dimensional Images*, 20:9–14, 2006.

[13] A. Mebarki, P. Alliez and O. Devillers. Farthest point seeding for placement of streamline. *Transaction on Visualization and Computer Graphics*, 10:479–486, 2005.

[14] H. Bhatia, S. Jadhav, V. Pascucci, G. Chen, J. Levine, L. Nonato and P.-T. Bremer. Flow visualization with quantified spatial and temporal errors using edge maps. *Transactions on Visualization and Computer Graphics*, 18(9):1383–1396, 2012.

[15] B. Preim and D. Bartz. *Visualization in medicine: theory, algorithms, and applications*. Morgan Kaufmann, 2007.

[16] R. Nascimento, J. Paixão, H. Lopes and T. Lewiner. Topology aware vector field denoising. In *Sibgrapi*, pages 103–109. IEEE, 2010.

[17] A. J. Smits and T. Lim, editors. *Flow visualization: techniques and examples*. Imperial College, 2 edition, 2012.

[18] X. Tricoche, G. Scheuermann and H. Hagen. Continuous topology simplification of planar vector fields. In *IEEE Visualization*, pages 159–166, 2001.

[19] X. Tricoche. Vector and tensor field topology simplification, tracking, and visualization. PhD thesis, *Schriftenreihe Fachbereich Informatik, University of Kaiserslautem*, 2002.

[20] G. Turk and D. Banks. Image-guided streamline placement. In *Siggraph*, pages 453–460, 1996.

[21] L.-Y. Wei. Visualizing flow fields by perceptual motion. Technical Report MSR-TR 82, Microsoft Research, 2006.