Eurographics Conference on Visualization (EuroVis) 2020
M. Gleicher, T. Landesberger von Antburg, and I. Viola
(Guest Editors)

Volume 39 (2020), Number 3

Feature Driven Combination of Animated Vector Field Visualizations

M-J. Lobo[1] , A.C. Telea[2] and C. Hurter[1]

[1]École Nationale de l'Aviation Civile
[2]Utrecht University, Netherlands

Abstract

Animated visualizations are one of the methods for finding and understanding complex structures of time-dependent vector fields. Many visualization designs can be used to this end, such as streamlines, vector glyphs, and image-based techniques. While all such designs can depict any vector field, their effectiveness in highlighting particular field aspects has not been fully explored. To fill this gap, we compare three animated vector field visualization techniques, OLIC, IBFV, and particles, for a critical point detection-and-classification task through a user study. Our results show that the effectiveness of the studied techniques depends on the nature of the critical points. We use these results to design a new flow visualization technique that combines all studied techniques in a single view by locally using the most effective technique for the patterns present in the flow data at that location. A second user study shows that our technique is more efficient and less error prone than the three other techniques used individually for the critical point detection task.

CCS Concepts

• ***Human-centered computing*** → ***Visualization techniques; Visualization design and evaluation methods;***

1. Introduction

Flow visualization has a long history with many techniques proposed to depict various aspects of vector fields [PVH*03, LHD*04, SLWS08, MLP*10, BCP*12]. These techniques critically use the human perception to support the detection and analysis of specific flow patterns. A subset of such techniques uses *animation* to show flow direction and velocity. In the last decades, many such animation techniques have been proposed, ranging from simple density plots [SS89] to GPU-accelerated 3D texture-based methods [TvW03].

While much effort has been invested in developing new flow visualization techniques, less was done to assess their effectiveness in terms of supporting understanding the flow data. Several studies have looked into *comparing* existing visualization techniques in a side-by-side setting to find the advantages and limitations thereof [LWSH04, LKJ*05]. Work has also been done to study how to *combine* different techniques to produce more insightful, easier to understand, visualizations [PVH*03, SK16]. The above two issues are related: To optimally combine different visualization techniques, we need to understand the types of flow patterns that each is best for. To our knowledge, no such study has been done for animation-based flow visualization techniques.

In this paper, we aim to bridge this gap as follows. First, we propose and execute a user study aiming to answer how *effective* different animation-based techniques are for the perception (visual detection and assessment) of various flow patterns such as critical points. The obtained insights show that different techniques perform differently for the studied flow pattern types. Based on these insights, we next propose a *composition* of the studied techniques driven by the flow *data* at hand. This way, the optimal technique (as found by our user study) is used in the flow region that contains the specific pattern it is best for. We smoothly combine the selected visualization techniques to yield the final global picture, so as to create an easy to interpret image where each blended visualization technique stands out where it is most needed. A second user study suggest that the composite technique produces less errors, and works faster, than the three techniques individually in a critical-point finding task.

We organize our work as follows. In Section 2, we overview related work on animation-based flow visualization. Section 3 presents the setup, execution, and conclusions of our first user study on the effectiveness of three animated flow visualizations for the detection of various types of simple flow patterns. Section 4 presents our approach to compose the studied visualization techniques based on the underlying flow data. Section 5 presents our second user study on assessing the effectiveness of the composite visualization. Section 6 presents several applications of our composite visualization for real-world flow data. Section 7 discusses the limitations of the composite visualization and the evaluation. Section 8 concludes the paper.

2. Related Work

Related work covers animation techniques for flow visualization, evaluating flow visualizations, and combining techniques in a single visualization, as follows.

2.1. Flow Visualization Techniques

Flow visualization techniques can be classified as direct [LHD*04], topology-aware [PVH*03], texture-based [LHD*04, LHZP07],

partition-based [SLWS08], geometric [MLP*10], and illustrative [BCP*12]. Our work relates to both animation-based and topology-aware techniques, so we discuss these in more detail next.

Animated techniques: Animation maps closely to the moving nature of flow data, so it can intuitively show flow direction, orientation, and magnitude. Image-based or texture-based techniques generate patterns that are locally aligned with the flow field and have a high spatial frequency orthogonally to the field, thus encode well the flow direction [CL93]. Generating time-dependent patterns by injecting and advecting noise or dye conveys next the flow speed and orientation, and can be efficiently implemented on the GPU using textures or shaders [MB95, vW02, vW03, LJH03, TvW03, LTH08, KSW*12]. Flow can also be visualized using motion maps that generate cyclical variable-speed animated textures [JL97, LJL04] or streamlines [JL00]. Several such techniques can also handle time-dependent flows, such as texture transport [BR98, JEH00, JEH01], IBFV [vW02], and Dynamic LIC (DLIC) [Sun03].

Besides textures and streamlines, other approaches use particles that move with the flow [SFL*04, ELPH19, BW08]. Particles create a high-contrast visualization that naturally conveys flow direction and speed (like textures) but create sparse visualizations (like streamlines).

Topology aware techniques: By analyzing the underlying vector field, one can detect special critical points or regions such as sources, sinks, vortices, laminar regions, and turbulent regions [HH89, LHZP07]. Such points or regions can be next explicitly visualized, used to decompose the flow field into a set of same-behavior regions [SS07], or used to control other visualizations such as placing streamline seed points [PVH*03, XLS10]. Topology methods, such as precise tracking of critical points over time, have also been proposed for time-dependent vector fields [TSH01].

2.2. Evaluating Vector Field Visualizations

Approaches to evaluate vector field visualizations can be divided into statistical methods and user studies. Statistical methods compare aspects of the actual (ground-truth) flow data, *e.g.*, orientation, to corresponding visual variables in the resulting (dense) visualization. If the latter is orientation, this can be computed by using the image gradient of the latter [MK13, JWC*11, PW08]. This approach has been used to compare different LIC parameter settings [JWC*11]. However, such measures address so far only static visualizations. In contrast, user studies compare different flow visualization by asking users to perform a specific task or to state their preference among several options. Results are based on control measures such as the error rate and/or the time needed to perform the task [PW13]. Such measures can be complemented with additional data such as eye tracking [HYL*15]. Earlier studies focus mainly on static visualization techniques [LCE*12, LKJ*05]. Ware *et al.* [WP13] also use eye-tracking to compare static flow visualizations on four experimental tasks. Interestingly, the tracking data shows that the users' gaze is mostly focused around critical points – so having a clear depiction of these is important. Later, Ware *et al.* [WBM*16] compared two animated techniques – orthogonal particles and oriented LIC (OLIC, [WGP97]) – to two static techniques – streamlets and arrow glyphs. Their results suggest that OLIC is better than the static techniques and orthogonal particles for some tasks. However, the two compared animated techniques, orthogonal particles and OLIC, differ in the information they show, so comparing them may not be fully fair. To our knowledge, there is no study comparing different animated techniques that differ in the representation density (sparse *vs* dense, that is) and that encode the same information (direction and orientation) of the vector field.

2.3. Combining Vector Field Visualizations

Flow visualization techniques have been combined to two ends. First, the earliest techniques already proposed to overlay multiple visualizations of the same flow data to show specific flow features (in focus) in the context of a more general, background-like visualization [TvW99, PVH*03]. This approach also helps visualizing multivariate flow-related data. Kirby *et al.* [KML99] show different flow attributes simultaneously using inspiration from painting. Schroeder *et al.* [SK16] propose an interface where users can overlay different visualizations interactively by sketching the region where the visualizations should be combined. Urness *et al.* [UIL*06] explore overlaying and embossing approaches to combine multiple vector fields. Verma and Pang [VP04] propose a visualization for the differences of two or more flow fields. Similar to some of these techniques, we also combine multiple visualizations to display a flow field. In contrast to them, however, we combine these *locally* and *automatically* based on where each is best suited to show the actual flow data, and use to this end *animation-based* techniques.

3. First User Study: Comparing Selected Visualizations

As outlined in Sec. 1, our first goal is to find out which animation techniques are best for common tasks related to vector field visual exploration. We first present the selection of considered techniques (Sec. 3.1). Next, we describe the evaluated tasks, study set-up, and obtained results (Sec. 3.2).

3.1. Chosen Visualization Techniques

Given Ware *et al.*'s study [WBM*16] that suggested that animated techniques are more effective than static ones for steady vector flows, we focus on animated techniques based on streamlines that can encode flow direction, orientation, and speed. For the case of time-dependent flows, these techniques will visualize the flow's streamline-oriented topology which is different from its pathline-oriented topology.

Additionally, we consider only techniques that can be computed in real time, to facilitate their usage in the user study described next. Following this search, we settled with three techniques, which are described next.

Image Based Flow Visualization (IBFV): This technique blends and deforms noise images in the flow direction [vW02]. Tuning its parameters can achieve a wide range of effects. IBFV is very simple to implement and works real-time using only plain OpenGL, which arguably contributes to its popularity. We render IBFV at 30 frames per second using WebGL. After experimentation, we found that the following parameters give good visualizations: 32 noise patterns of 256^2 pixels resolution; visualization resolution: 512^2 pixels; and maximum displacement at each iteration: 2 pixels. These values are quite close to those proposed originally [vW02].

Oriented Line Integral Convolution (OLIC): OLIC [WGP97] is a variant of LIC [CL93] where a noise texture is convolved in the flow direction. While IBFV uses dense noise, OLIC uses a sparse noise texture and a ramp-like kernel describing noise injection, producing variable-intensity 'droplets' that follow the flow. OLIC also proposed cycling through the kernel to produce an animation where the lighter

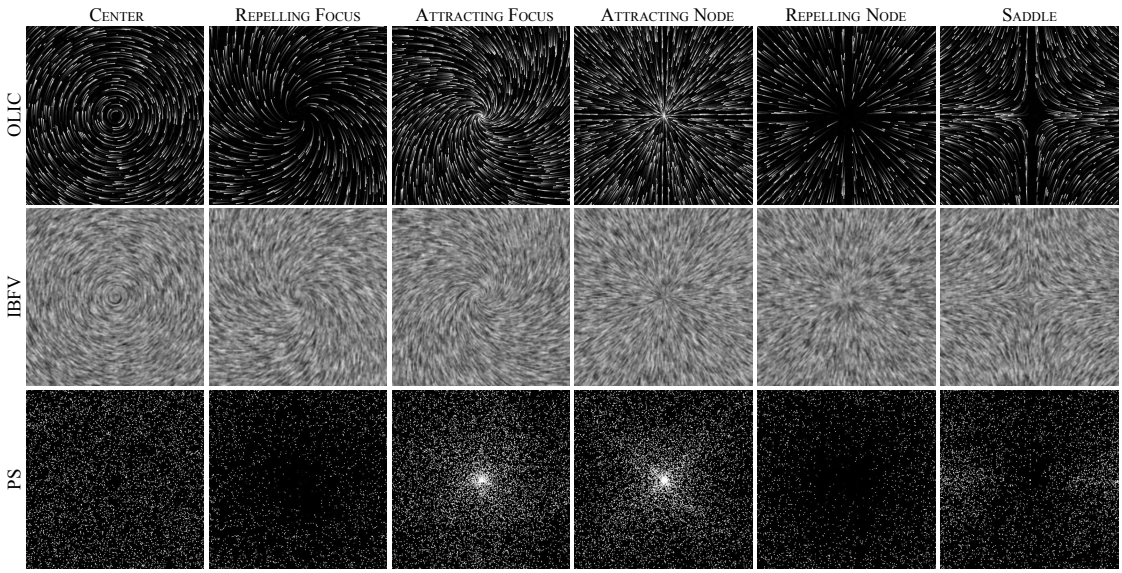

Figure 1: *Rendering each* pattern *with the three studied* techniques. *See Sec. 3.2.*

point of the droplet moves in the direction and orientation of the flow. OLIC was extended by FROLIC [WG97] to use a color table approach to speed up the animation. Our implementation, written in Javascript Canvas, follows FROLIC. We seed 2000 droplets by randomly jittering seed positions given by an uniform grid covering the visualization area. We trace forward and backward streamlines seeded at each droplet position and render them with a basis length of 50 pixels, scaled next by flow magnitude [WGP97], and thickness of 2 pixels. We decrease the droplets' opacities linearly along streamlines from a maximum value that maps the flow magnitude, so more opacity and/or longer droplets indicate faster flow.

Particle Systems (PS): This technique is one of the oldest in flow visualization [Bun89]. Yet, it is attractive as it creates high-contrast images with limited clutter, allows precise control of where information (particles) is drawn, is simple to implement, and is very fast [KKKW05]. A set of particles is advected over time in the flow field and drawn as a point cloud. Each particle has an initial position and a lifetime. When a particle exceeds its lifetime, stops moving (because it reaches a critical point), or exits the dataset domain, we re-seed its position at a random location. To obtain a roughly uniform particle density over similar-divergence areas of the flow domain, we seed more particles in high-speed flow areas. For this, we define the probability $p(\mathbf{v})$ for a particle to be seeded at a location having the vector value $\mathbf{v}$ as

$$p(\mathbf{v}) = (2p_t - 1)\frac{\|\mathbf{v}\|}{\|\mathbf{v}_{max}\|} + 1 - p_t, \tag{1}$$

where $\mathbf{v}_{max}$ is the largest vector in the field. Next, we release particles according to these probabilities. The factor $p_t \in [0,1]$ controls the overall particle density – higher values generate denser particle plots. Note that this does *not* aim to create an *overall* uniform particle density (as in *e.g.* DLIC [Sun03]). For that, more complex and expensive schemes would be needed to estimate the particle density and adjust it locally by adding or removing particles. Also, such an uniform seeding would not be ideal for our usage of PS, as finding certain patterns such as attracting and repelling foci (discussed in the next section, and shown in Fig. 1, bottom row).

3.2. User Study

To compare the effectiveness of the three selected visualization methods (IBFV, OLIC, PS), we next designed and executed an user study.

Tasks: Previous work has shown that detecting and understanding critical points is an important requirement of flow visualizations [WP13]. Hence, our *tasks* in the study cover finding and classifying various types of *critical points* present in the flow data. This task has been widely used in the literature to compare flow visualization techniques [LKJ*05, LCE*12, WP13] because it requires inspecting the flow both globally and locally [LCE*12] and thus represents well any task that requires this type of understanding.

Data: As ground-truth for critical point finding and classification, we use a synthetic vector field. Following [LCE*12], we want to explicitly define the number and type of critical points present in a field so as to remove potential biases due to using only one vector field, but keep complexity across vector fields similar. We synthesize vector fields $\mathbf{v} : \mathbb{R}^2 \to \mathbb{R}^2$ with a potential model [vW02] given by

$$\mathbf{v}(\mathbf{x}) = k \begin{bmatrix} a & b \\ c & d \end{bmatrix} \frac{\mathbf{d}}{\|\mathbf{d}\|^2}, \tag{2}$$

where $\mathbf{d} = \mathbf{x} - \mathbf{c}$ is the position of point $\mathbf{x}$ with respect to the visualization center $\mathbf{c}$ and k controls the overall vector field speed. The coefficients a, b, c, d control the creation of different flow patterns (Tab. 1). We randomly choose clockwise or anticlockwise rotations for CENTER, ATTRACTING FOCUS, and REPELLING FOCUS for each trial. We compose the effects of multiple critical points by summing up their corresponding fields given by Eqn. 2. We synthesize fields using 5 types of generators (instances of Eqn. 2) for centers, attracting and repelling nodes, and attracting and repelling foci respectively. Saddles are created implicitly by the interaction of the other generators. Figure 1 shows all generators rendered by OLIC, IBFV, and PS.

To create the fields, we define a 5×5 grid that divides the domain in equally sized cells. In each vector field, we add 3 instances of each of the 5 generator types by randomly picking a grid cell and adding the corresponding generator to its center, followed by a random jitter

Pattern type	Parameter settings
Center	$a = d = 0, b = c$
Attracting node	$a = -d, a < 0, b = c = 0$
Repelling node	$a = -d, a > 0, b = c = 0$
Saddle	$a = d, b = c = 0$
Attracting focus (anticlockwise)	$a = b = c - d, a < 0$
Attracting focus (clockwise)	$-a = b = c = d, a < 0$
Repelling focus (anticlockwise)	$a = b = c = -d, a > 0$
Repelling focus (clockwise)	$-a = b = c = d, a > 0$

Table 1: *Parameters for generating the different flow patterns.*

of half a cell size. We add only one generator per cell to spread critical points over the visualization. Next, we use a standard fixed-point classification algorithm (for details, see Sec. 4) to find if saddles were created by these generators. If so, we count these up. We repeat the process until exactly 10 saddles have been created. This yields thus a total of $3 \times 5 + 10 = 25$ critical points per field, one in each grid cell.

Participants: Fifteen unpaid volunteers (3 females, 12 males), daily computer users, aged 23 to 56 (mean 32.6, median 27), not involved in this research, participated in the study. All had normal or corrected-to-normal vision, did not suffer from color blindness, and had previous knowledge of vector fields.

Hypotheses: An earlier related study [LCE*12] showed that EnhancedLIC, a texture-based static visualization technique for flow data, outperformed OLIC for critical point recognition and classification. The study also suggests that the higher density of Enhanced LIC was of important added value in better understanding the flow topology. Following this, we hypothesize in our case that:

H1: Since the sparse visualizations (OLIC, PS) are less dense around positive-divergence points (REPELLING FOCUS and REPELLING NODE) than elsewhere, detecting these two kinds of points will be harder with OLIC and PS than with IBFV (which has an uniform and high density everywhere).

H2: For the same reason, critical points that attract flow (ATTRACTING FOCUS and ATTRACTING NODE) will be easier to find using the two sparse techniques (OLIC, PS) than with IBFV, since OLIC and PS generate higher-density patterns around ATTRACTING FOCUS and ATTRACTING NODE than elsewhere.

H3: As SADDLE points both repel and attract flow, they should be easier to find using OLIC because this technique maps both orientation and direction explicitly into its high-contrast droplets. At the other end, PS encodes direction via the harder-to-gauge points animation, so it will be the least suitable technique for finding saddles. IBFV encodes direction in texture patterns which are lower-contrast than OLIC's droplets, but more explicit than the implicit point-movement direction of PS. Hence, IBFV should score in-between OLIC and PS for finding saddles.

Procedure: We follow a 3×7 within-subject design with 2 factors: *technique* and *pattern*. *Technique* covers IBFV, OLIC, and PS. *Pattern* covers the 5 generator types CENTER, REPELLING FOCUS, ATTRACTING FOCUS, REPELLING NODE, ATTRACTING NODE, plus the implicitly created SADDLEs. Each condition was repeated three times. Trials were grouped by *technique* and *pattern* blocks. The presentation order of *techniques* was counterbalanced across participants. For each block, the order of *patterns* was randomized. For each *pattern*, a training trial preceded the 3 actual trials. We generated 9 synthetic fields as described earlier, with normalized speed, plus one for the first training. Each subset of three fields from these 9 was used for every *pattern* and for each *technique* combination, for the three actual trials. For the second and third training, we re-used one of the previous fields. We presented the fields in the same order to each participant, since the order of the other factors was randomized. Visualizations were shown at 512^2 pixel resolution in a web browser.

For each trial, participants were asked to click on locations where they see the three critical points for the indicated *pattern*. Upon clicking within a tolerance under half a cell size to the correct location, a green square was shown at the actual critical-point location to tell that the answer was correct. Clicking at an incorrect location (not corresponding to the target *pattern*) had no effects, thereby indicating the need to retry. Upon finding all three target points, or when a 40 seconds timeout elapsed, the trial ended and the next one started.

Results: We base our analysis on 95% bias-corrected and accelerated (BCa) bootstrap confidence intervals [Dra16], using 10000 replications. Confidence intervals have been used in many recent HCI and visualization studies [BBB*19, PAPB19], and have been preferred over p-values as the latter may lead to dichotomous thinking [BD17]. We compare techniques by calculating the confidence intervals of their pairwise difference [Cum14] adjusted for multiple comparisons using Bonferroni correction. We say that there is strong evidence for a difference when the confidence interval does not contain the zero value. All material (data logs, R scripts) is available online [The19].

Figures 2 and 3 show the number of errors (critical points wrongly classified), respectively the mean time for finding a correct critical point, per *pattern* and per *technique*, using 95% confidence intervals. For completion time, we excluded trials where participants did not find at least one critical point before the timeout. Overall, the three techniques appear to have comparable performance. However, looking at each (*technique*, *pattern*) condition, several differences become visible, as follows. For some *patterns*, there is strong evidence that techniques perform quite differently: For ATTRACTING FOCUS and ATTRACTING NODE, OLIC and PS outperform IBFV roughly equally. For REPELLING NODE, PS and IBFV outperform OLIC, for REPELLING FOCUS IBFV seems to be the fastest technique. For SADDLE both OLIC and IBFV outperform PS, and for CENTER there is evidence that IBFV is fastest than OLIC. A correlated pattern is visible in the number of errors: For SADDLE, IBFV and OLIC outperform PS; and for CENTER, IBFV seems to be better than PS, and PS outperforms OLIC. For REPELLING FOCUS there is strong evidence that IBFV outperforms OLIC and PS.

The results of this study, both considering error rates and completion times, show that some techniques work better for detecting some critical point types than others. Our first hypothesis (H1) is thus supported: IBFV seems to be the most effective to detect REPELLING NODE and REPELLING FOCUS. Our second hypothesis (H2) is also supported, as OLIC and PS are more efficient for finding ATTRACTING NODE and ATTRACTING FOCUS points than IBFV, and there is no significant difference between them. Our third hypothesis (H3) is only partially supported: OLIC is better than PS for finding SADDLE points, but so is IBFV. This suggests that the density of the visual representation is a more important factor than displaying both orientation and direction both statically and dynamically.

4. Compositing flow visualizations

Our first study (Sec. 3.2) suggests that some of the studied techniques (IBFV, OLIC, Particles) better convey certain critical point types than others. Hence, it is defensible to consider a visualization that

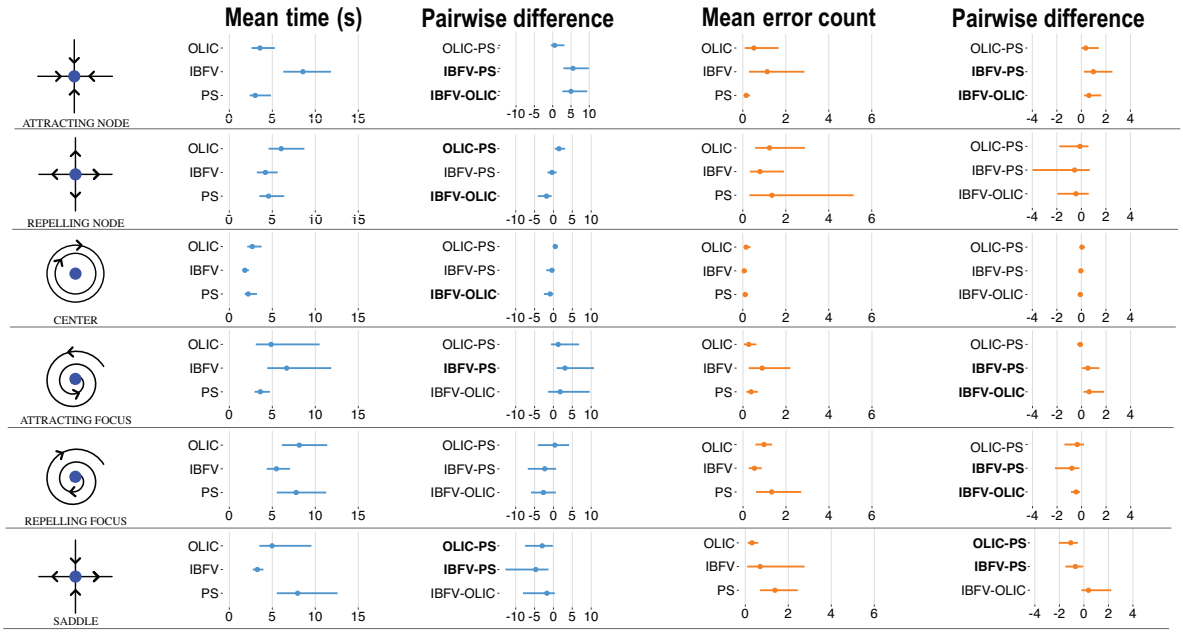

Figure 2: *Mean time, error count, and pairwise differences per* TECHNIQUE *and* PATTERN. *Bootstraped 95% confidence intervals.*

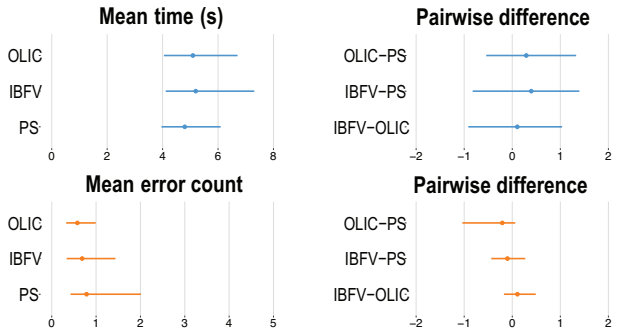

Figure 3: *Mean error count, time and pairwise difference per* TECHNIQUE.

Point type	Real part	Imaginary part
Saddle	$R1 < 0, R2 > 0$	$I1 = I2 = 0$
Repelling Node	$R1 > 0, R2 > 0$	$I1 = I2 = 0$
Attracting Node	$R1 < 0, R2 < 0$	$I1 = I2 = 0$
Attracting Focus	$R1 = R2 < 0$	$I1 = -I2 \neq 0$
Repelling Focus	$R1 = R2 > 0$	$I1 = -I2 \neq 0$
Center	$R1 = R2 = 0$	$I1 = -I2 \neq 0$

Table 2: *Classifying critical points.*

composes these techniques so that they become prominent in areas where they work best, based on the presence of those patterns they can best depict. We explore this design next (see also Fig. 4).

4.1. Composition design for static vector fields

Given an arbitrary steady 2D vector field $\mathbf{v}(\mathbf{x}) = (u(\mathbf{x}), v(\mathbf{x}))$, we first find critical points looking for locations where the u and v components change signs. We next classify the found points using the eigenvalues of the Jacobian matrix of $\mathbf{v}$

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial u}{\partial x}(\mathbf{x}) & \frac{\partial u}{\partial y}(\mathbf{x}) \\ \frac{\partial v}{\partial x}(\mathbf{x}) & \frac{\partial v}{\partial y}(\mathbf{x}) \end{bmatrix}. \tag{3}$$

Following [HH89], the real $(R1, R2)$ and imaginary $(I1, I2)$ parts of the two eigenvalues can discriminate between critical points (Tab. 2). For numerical robustness, we perform all above comparisons with a tolerance $\varepsilon = 0.1$. To choose this value, we tested values from 0.01 to 0.2 and chose the one that resulted in the most accurate detection. This simple critical point detection is very fast, which is essential to create animated visualizations for time-dependent vector fields. More advanced methods can be used, if desired, for critical point detection [LHZP07]. Note that our method is robust with respect to this step: If some critical points are not detected, the composite visualization will *still* show all the field data, though possibly using a suboptimal technique in areas where detection failed. This is in contrast to vector field topological analysis where *accurate* and *complete* detection of such points is paramount.

After detecting N critical points at locations $\mathbf{x}_1, \ldots, \mathbf{x}_N$, we create the N visualizations (images) $V_1, \ldots, V_N$ that best match the types of $\mathbf{x}_i$ following Sec. 3. Next, we compose these in a final image V by weighting them, at each pixel $\mathbf{x}$, by the distance from $\mathbf{x}$ to $\mathbf{x}_i$, *i.e.*

$$V(\mathbf{x}) = \begin{cases} \frac{\sum_{i=1}^{N} w_i(\mathbf{x}) \cdot V_i(\mathbf{x})}{\sum_{i=1}^{N} w_i(\mathbf{x})} & \text{if } \|\mathbf{x} - \mathbf{x}_i\| \neq 0 \text{ for all } i \\ V_i(\mathbf{x}) & \text{if } \|\mathbf{x} - \mathbf{x}_i\| = 0 \text{ for some } i \end{cases} \tag{4}$$

Here, $w_i(\mathbf{x}) = 1/(\|\mathbf{x} - \mathbf{x}_i\| \cdot f(t_i))^p + f'(t_i)$ are inverse-distance (Shepard [She68]) weight functions (weight maps). The term $f'(t_i)$ ensures that all visualizations V_i are still partially visible even far away from their respective critical points $\mathbf{x}_i$. The power p controls the weighting function decay. We tested different p values for the Shepard interpolation. Higher values result in a more clear distinction between the different critical point areas, but the transition between visualizations is less smooth. We settled for $p = 2$ as this provides a good trade-off between the smoothness of the visualization and the visibility of the different visualization regions. The term $f(t_i)$

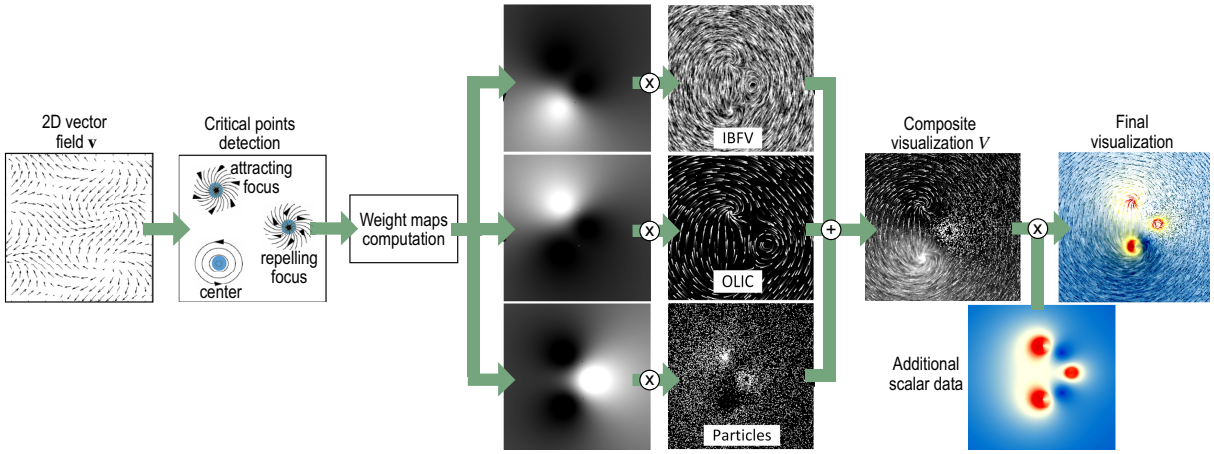

Figure 4: *Pipeline for the visualization compositing process. See Sec. 4.*

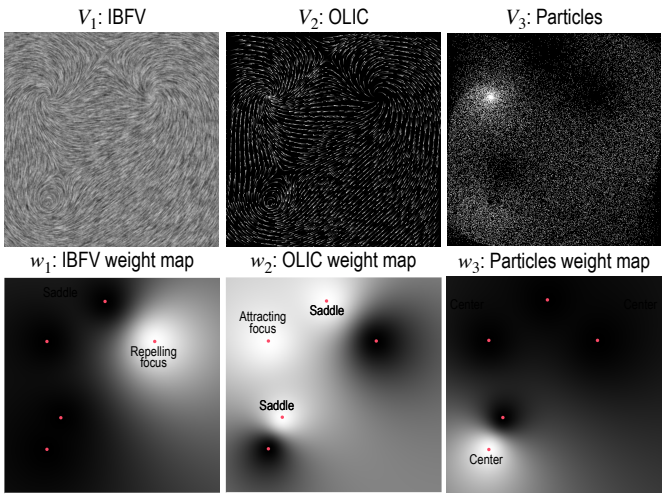

Figure 5: *Visualizations V_i and corresponding weight maps w_i computed from a real vector field. Bottom row also shows the location and type of critical points. See Sec. 4.1.*

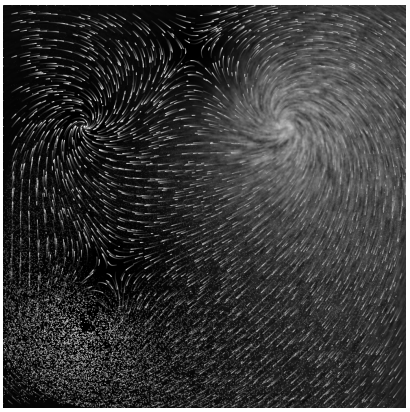

Figure 6: *Composite visualization created by assigning IBFV to* REPELLING FOCUS *and* REPELLING NODE, *OLIC to* ATTRACTING FOCUS *and* SADDLE, *and PS to* CENTER *and* ATTRACTING NODE. *See the individual IBFV, OLIC, and PS visualizations in Fig. 5*

controls the salience of the visualization V_i. Figure 6 is generated using the default settings $f'(t_i) = 1$, $p = 2$, $f(t_i) = 1$ that we propose for the composition. The corresponding visualizations V_i and weight maps w_i for this figure are shown in Fig. 5. For presentation simplicity, we merged here all visualizations of the same type (IBFV, OLIC, and Particles), assigned to the $N = 25$ found critical points, into three images (top row in Fig. 5). Comparing the composite result (Fig. 6) with the three individual visualizations (Fig. 5, top row), we argue that the composition shows the structure of the underlying field much more clearly.

Changing the parameter f', p, and f allows (de)emphasizing certain visualizations so as to reduce the visual disparity between them and make them equally salient for the user: For example, IBFV has an overall higher brightness than OLIC or Particles, so one would arguably like to de-emphasize the former and/or emphasize the latter two. Figure 4 illustrates this. Here, we weaken the visual salience of saddles (assigned to IBFV) by using a higher factor $f($ SADDLE$)$ than for the other critical points.

After composition, we can use the so-far unexploited color channel to encode supplementary data. Figure 4 (rightmost image) illustrates this by adding the flow magnitude color-coded using a blue-yellow-red colormap. The same technique is used when generating the visualizations discussed next in the application part (Sec. 6). Our implementation uses Javascript and WebGL. On a MacBook Pro Retina 15" equipped with an NVIDIA GeForce GT 750M 2048 MB, it can render 30 frames per second at a resolution of 1024*1024 pixels.

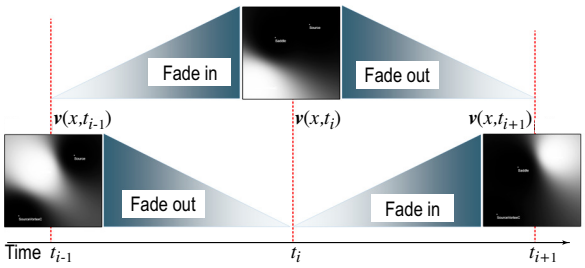

Figure 7: *Compositing visualizations for time-dependent flow fields.*

4.2. Visualizing time-dependent fields

Our visualization compositing can be adapted to handle time-dependent flow data. We consider such datasets as sequences of 2D vector fields $\mathbf{v}(\mathbf{x},t_j)$ recorded at time moments t_j. For each sequence frame, we construct its composite visualization $V(\mathbf{x},t_j)$ following Eqn. 4.

Directly drawing the sequence $V(\mathbf{x},t_j)$ however yields a distracting animation since, at each t_i, the weights of the three individual visualization techniques V_i in Eqn. 4 can change significantly, and so does the vector field itself. To address this, we create a temporal blending that linearly fades out the current frame $V(\mathbf{x},t_j)$ while fading in the new frame $V(\mathbf{x},t_{j+1})$ over a user-set duration of $\tau = 3$ seconds (Fig. 7, see also the companion video). The parameter τ controls the speed of the animation. Similar techniques have been used to create smooth transitions in time-dependent visualizations for various data such as time-dependent graphs [HET13], scalar data cubes [HTCT14], and – closest to our scope – vector fields depicted by dense textures [vW02].

5. Second User Study: Assessing Visualization Composition

The compositing idea in Sec. 4 hypothesizes that combining different visualizations of vector data according to underlying critical points is more effective than visualizing the same data by each technique individually. To test this, we conducted a second user study that compared the OLIC, IBFV, and PS techniques to the compositing technique (called next MIX).

Tasks: Our task is the same as in the first user study (Sec. 3.2), *i.e.*, finding critical points in a visualized flow field.

Design: We also follow the design of the first study. We generate 16 vector fields using the potential model (4 for training and 12 for actual trials), with the same restrictions for critical points. We next use the results of the first study to choose which visualization to assign to a critical point, as follows. To have a balance between the composited visualizations, we assign two types of critical points to each of them. For this, we look at each *pattern* individually. If the first study found that a technique V_i is more effective or efficient than another one V_j, we use V_i for that *pattern*. Otherwise, we choose the technique that produced the lowest error count. This results in using OLIC for ATTRACTING FOCUS and SADDLE, IBFV for REPELLING FOCUS and REPELLING NODE, and PS for ATTRACTING NODE. Since we want to have two *patterns* per technique, we use PS for CENTER, since PS produces similar results to the other techniques, even if IBFV produces a lower error count. Figure 8 shows a visualization created using this mapping of techniques to patterns.

Hypothesis: MIX assigns the techniques that work best for each *pattern*, so we expect this MIX to be both more efficient and effective than the individual techniques (OLIC, IBFV, PS) for the critical point finding task, *i.e.*, yield a lower error count and mean time.

Participants: Sixteen unpaid volunteers (3 females, 13 males), daily computer users, aged 21 to 45 (mean 30.8, median 27), not involved in this research, participated in the study. All had normal or corrected-to-normal vision, did not suffer from color blindness, and were familiar with general tasks in vector field visualization.

Procedure: We followed a 4×6 within-subject design with 2 factors: *technique* and *pattern*. *Technique* covers, again, IBFV, OLIC, Particles, and MIX. *Pattern* covers, again, the 5 generator types CENTER, REPELLING FOCUS, ATTRACTING FOCUS, REPELLING NODE, ATTRACTING NODE, plus the implicitly created SADDLEs. We follow the same procedure as in the first study, using now 16 vector fields (4 for training and 12 for trials). For this study, we also asked participants to rank the perceived difficulty of each TECHNIQUE * PATTERN combination after each trial block.

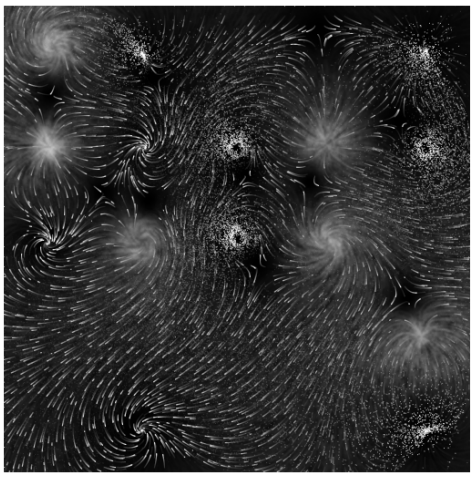

Figure 8: *Composite visualization from the second study (Sec. 5).*

Results: Figure 10 shows the time and number of errors per TECHNIQUE, with their 95% bootstraped confidence intervals. The differences between technique-pairs provide strong evidence that MIX is faster than the three other techniques (64 % faster than PS, 68 % than IBFV and 78 % than PS), because the three confidence intervals of the differences do not contain the zero value. Also, we see evidence that MIX produces fewer errors than OLIC and PS (140 % fewer errors than PS, 50 % fewer errors than IBFV, and 135% fewer errors than OLIC). Hence, our hypothesis is confirmed. We see similar results for the perceived difficulty of each technique (Fig. 10).

Figure 9 depicts the results per *pattern*. The pairs for which there is strong evidence of a difference between techniques have bold names. These results match the previous one: IBFV performs worst than the other techniques for ATTRACTING NODE and ATTRACTING FOCUS; PS is slower than IBFV and OLIC for SADDLE; and OLIC is slower than PS for REPELLING NODE. We also find evidence that MIX is faster than IBFV for all patterns except CENTER; faster than OLIC for REPELLING NODE, REPELLING FOCUS and SADDLE; and faster than PS for REPELLING NODE, REPELLING FOCUS and SADDLE. Looking at the error count, we see that MIX produces fewer errors than OLIC when finding ATTRACTING NODE, REPELLING NODE, CENTER and REPELLING FOCUS; fewer errors than IBFV when finding ATTRACTING FOCUS and REPELLING FOCUS; and fewer errors than PS when finding REPELLING FOCUS and SADDLE.

6. Applications

We now demonstrate our composite visualization on two real-world datasets and use-cases.

6.1. Use case 1: Scientific Visualization

We first consider a wind-flow dataset containing actual measured wind direction and magnitude (speed), measured at different times and altitudes. The dataset, provided by Météo-France, is 587×625 nautical miles (NM), covering the French territory, with altitudes

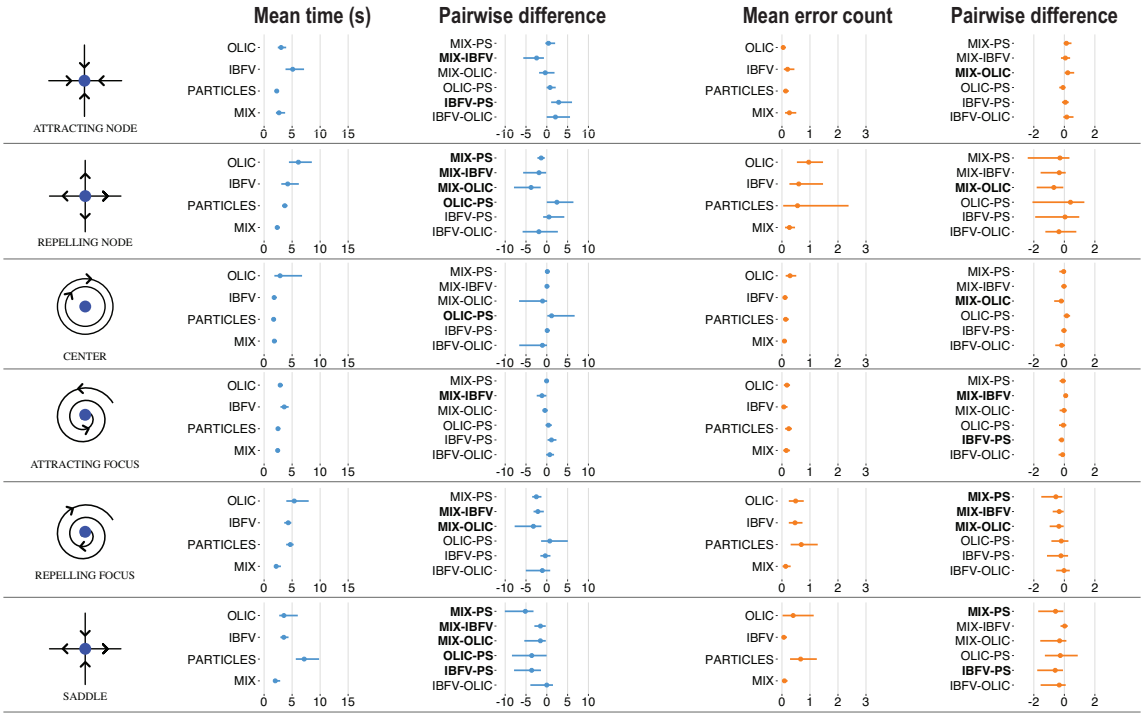

Figure 9: *Mean error count, time, and pairwise differences per* PATTERN *per* TECHNIQUE. *Bootstraped 95% confidence intervals.*

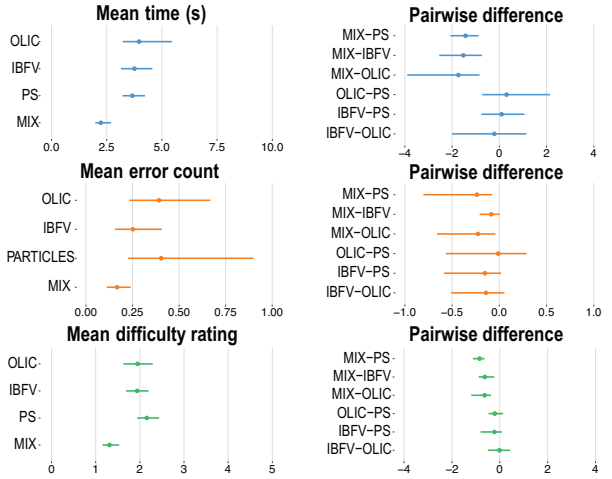

Figure 10: *Time, error count, and rated difficulty per* TECHNIQUE. *Bootstraped 95% confidence intervals.*

ranging from 110.6 meters to 10363 meters. Data consists of 8 time steps, each measured every 3 hours. Per timestep, data is spatially sampled over an uniform grid of 150 $(x) \times 100$ $(y) \times 10$ $(height)$ cells. 2D slices of such (and similar) datasets are used to understand how wind direction evolves over time, how it varies from lower to higher altitudes and to find patters such as cyclones and anticyclones [WP13], as required for weather analysis and forecasting [MSIMI*08]. We present next two uses of our composite visualization for this dataset.

Wind layer exploration: Exploring this wind dataset has been previously done with small multiples, each for a specific layer [HAG*14]. The problem with this approach is that vertical or horizontal moving patterns are not easy to see or trace. Another possibility is to use 3D streamlines, isosurfaces, and/or volume rendering [RGG19]. However, this brings well-known problems of choosing suitable viewpoints, occlusion, and many parameters to tune. In contrast, our technique shows a single (animated) dense 2D visualization (Fig. 13). To achieve this, given a user-chosen time-of-the day t_0, we take the 2D time-dependent field $\mathbf{v}(x,y,h)$ where $\mathbf{v}(x,y,t)$ is a slice at height z in the original field $\mathbf{v}(x,y,z,t_0)$, and next visualize it with the time-dependent composite technique described in Sec. 4.2. Thanks to this technique, we can smoothly transition between layers to see how the wind data gradually changes with altitude, to form a mental view of the weather patterns. Figure 13 and the accompanying video show that low altitude flow is more complex with many local flow pattern types; high altitude flow is sparser but with stronger wind magnitude. Our animation also shows how the wind's main direction turns when altitude changes, rotating clockwise in the Northern hemisphere (our case) and anticlockwise in the Southern one, a phenomenon known as the Bernoulli effect [Tru84]. Standard slice-based visualizations as well as static 3D flow visualizations fail to emphasize this effect.

Time-dependent exploration: In this second scenario, we are interested to examine data over time. Since our composite visualization is 2D, we limit ourselves to a single altitude layer (4 km height), but consider all time frames. An important task for this type of weather data is to find out where, and when, do specific patterns (such as critical points) appear or disappear. Doing this using a standard animation based on a *single* visualization technique may be hard, since the strength and spatial extent of such changes can be limited. Our composite technique helps here since, as critical points appear or disappear, this triggers the blending (or removal) of a *different* type of visualization around those locations. Figure 11 and the accompanying video illustrate this. In the first frame, taken at 6 AM, we spot an

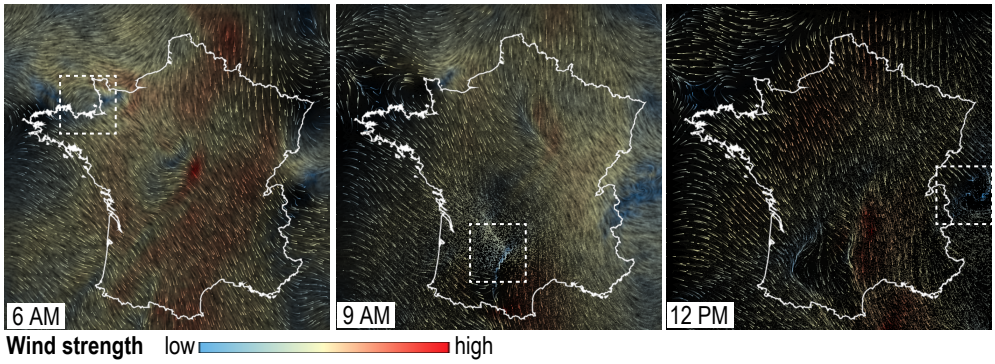

Figure 11: *Wind field at 4 km altitude from 6 AM to 12 AM. Emerging patterns are highlighted. See also companion video.*

ATTRACTING FOCUS in the top left, depicted using IBFV. At 9 AM, this critical point disappears and an ATTRACTING NODE emerges at the bottom of the field, depicted using PS. The corresponding fading out of IBFV, and the fading in of PS, attracts attention to these regions and thus makes it easier to detect the changes. A similar situation happens between the second and third frame where the PS at the bottom fade out and an ATTRACTING FOCUS region appears to the right, depicted with PS. The appearance of a new PS area at this place attracts the viewer's attention, making its discovery easier.

6.2. Use case 2: Information Visualization

Simplified visualization of spatial trails has a long history in information visualization [LHT17, Hur15]. One of the techniques for this is *bundling*, which groups spatially-close trails into simpler structures to reduce clutter and expose key patterns in the input trails. State-of-the-art bundling techniques model bundling as an advection of the trails in a suitably designed vector field $\mathbf{v}$. Setting $\mathbf{v}$ to different fields yields variations such as isotropic bundling [HET12] or the more powerful attribute-driven edge bundling (ADEB, [PHT15]). Controlling such techniques to yield the desired number of bundles and/or separation of different bundles, is very challenging [vdZCT16].

We next show how visualizing the field $\mathbf{v}$ helps this process for ADEB. We first outline how $\mathbf{v}$ is generated. Let $T = \{e_i\}_{1 \le i \le N} \subset \mathbb{R}^2$ be a set of N 2D trails, each being represented as a sampled curve. A trail density map $\rho : \mathbb{R}^2 \to \mathbb{R}^+$ is computed by kernel density estimation as

$$\rho(\mathbf{x} \in \mathbb{R}^2) = \frac{1}{h^2} \sum_{i=1}^{N} \int_{\mathbf{y} \in e_i} K\left(\frac{\|\mathbf{x}-\mathbf{y}\|}{h}\right), \tag{5}$$

where $K : \mathbb{R}^2 \to \mathbb{R}^+$ is a symmetric 1D non-negative kernel of bandwidth $h > 0$, *e.g.*, Gaussian or Epanechnikov. ADEB [PHT15] defines a *flow direction map* $\theta : \mathbb{R}^2 \to \mathbb{R}^2$

$$\theta(\mathbf{x} \in \mathbb{R}^2) = \frac{1}{h^2} \sum_{i=1}^{N} \int_{\mathbf{y} \in e_i} \mathbf{d}(\mathbf{y}) K\left(\frac{\|\mathbf{x}-\mathbf{y}\|}{h}\right). \tag{6}$$

which is a vector field that extrapolates the tangent vectors $\mathbf{d}(\mathbf{x})$ at all edge sampling points $e_i(\mathbf{x})$ to the entire 2D domain. Using θ, we define at each point $\mathbf{x} \in T$ a subspace of *compatible directions* as

$$\Omega_{\mathbf{x},c} = \left\{ \mathbf{y} \in \mathbb{R}^2 \setminus \ker(\theta) \;\middle|\; \frac{\mathbf{d}(\mathbf{x}) \cdot \theta(\mathbf{y})}{\|\theta(\mathbf{y})\|} \ge c \right\} \subset \mathbb{R}^2. \tag{7}$$

The ADEB bundling of T is now the fixed point of the ordinary differential equation

$$\frac{d\mathbf{x}}{dt} = \frac{h(t) \nabla_{\Omega_{\mathbf{x},c}} \rho(\mathbf{x},t)}{\max(\|\nabla_{\Omega_{\mathbf{x},c}} \rho(\mathbf{x},t)\|, \varepsilon)}, \quad \mathbf{x} \in T. \tag{8}$$

Controlling isotropic bundling is well understood, as detailed analyses of all its parameters exist [HET12, vdZCT16]. ADEB (Eqn. 8) is much harder to control to obtain the desired bundles, and no similar parameter analysis exists for it. The key problem here is to understand how the tangent field θ (Eqn. 6) influences the gradient $\nabla_{\Omega_{\mathbf{x},c}} \rho$ used in Eqn. 8 to advect trails. To help this understanding, we visualize θ with our composite method. Using animation makes sense here since θ is a vector field in which the trail-set T is literally *advected* during bundling. Also, we know that sources and sinks of such advection fields strongly influence bundling results [EHP*11]. Hence, we argue that studying such points for the complex field θ will give us insights into the behavior of ADEB.

Figure 12 shows four visualizations of the field θ, with flow magnitude encoded by color, and the trail-set drawn atop. We first show the original, unbundled, trail set (a). This contains 22720 flight paths of airplanes during February 8, 2009, over the French air space (for details, see [PHT15]). Here, the flow is strongest over a concentrated area over the Paris region. The main trail-groups are also visible (orange bands in the image). This is an important hint that bundling can handle well this dataset. Indeed, if these trail-groups had a similar field magnitude surrounding them, there would be too weak advection to bundle them. Image (b) shows the effect of 10 ADEB bundling iterations with an *average* kernel $h = 10$ pixels. The obtained bundling shows a good balance of details *vs* simplification, and a field θ that is much more complex than the original one. We see how θ drives bundling – inset shows how the main bundles align with the white 'trails' formed by OLIC droplets. Besides being well aligned with bundles, θ is also strong along these and weak elsewhere. This tells that bundling has *converged*, so no more bundling iterations are needed. Image (c) shows the effect of using a *large* kernel $h = 60$: θ changes drastically – it is simpler, smoother, and has some large vortices inside empty areas surrounded by the emerging bundles. Indeed, a larger kernel extrapolates the trails' tangent directions far away in space (Eqn. 6), so it is natural for such vortices to appear between bundles. This field creates however a too simplified bundling. At the other extreme, image (d) uses a *small* kernel $h = 6$. Now θ is almost zero outside the unbundled trail set. The resulting bundling is very weak being actually similar to the original unbundled trails (a). We also verified that further decreasing the kernel immediately creates convergence problems for the advection (Eqn. 8). The direction fields

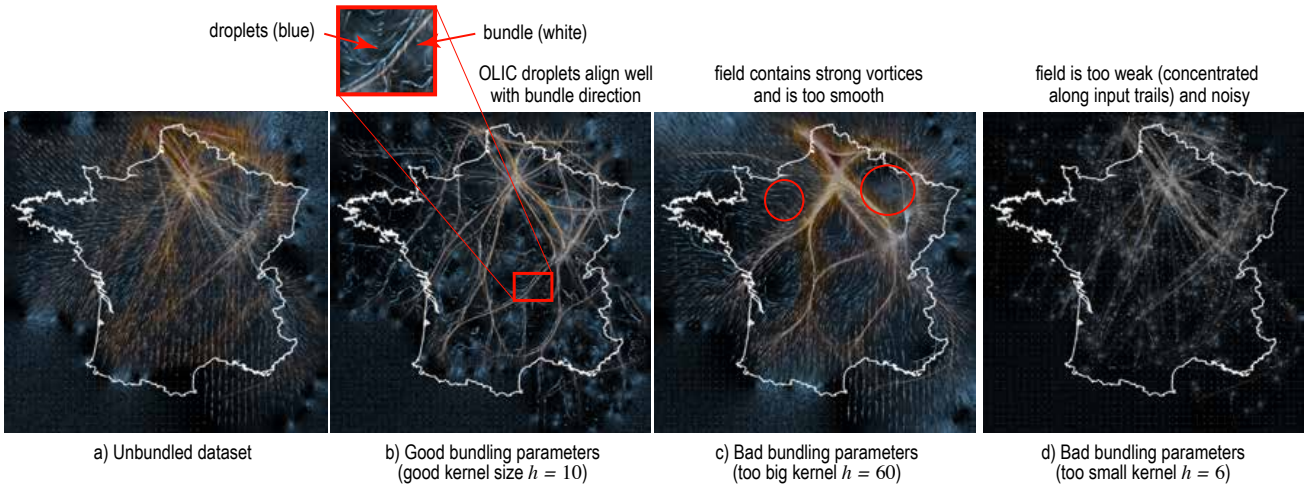

Figure 12: *Flow direction field visualization for ADEB. (a) Original (unbundled) trail set and corresponding field at the beginning of the bundling. (b) Result of 10 bundling iterations with a good kernel size. (c,d) Result of bundling for badly chosen (too large, respectively too small) kernel sizes. Note how the bundling level-of-detail is well reflected in the complexity of the direction flow field.*

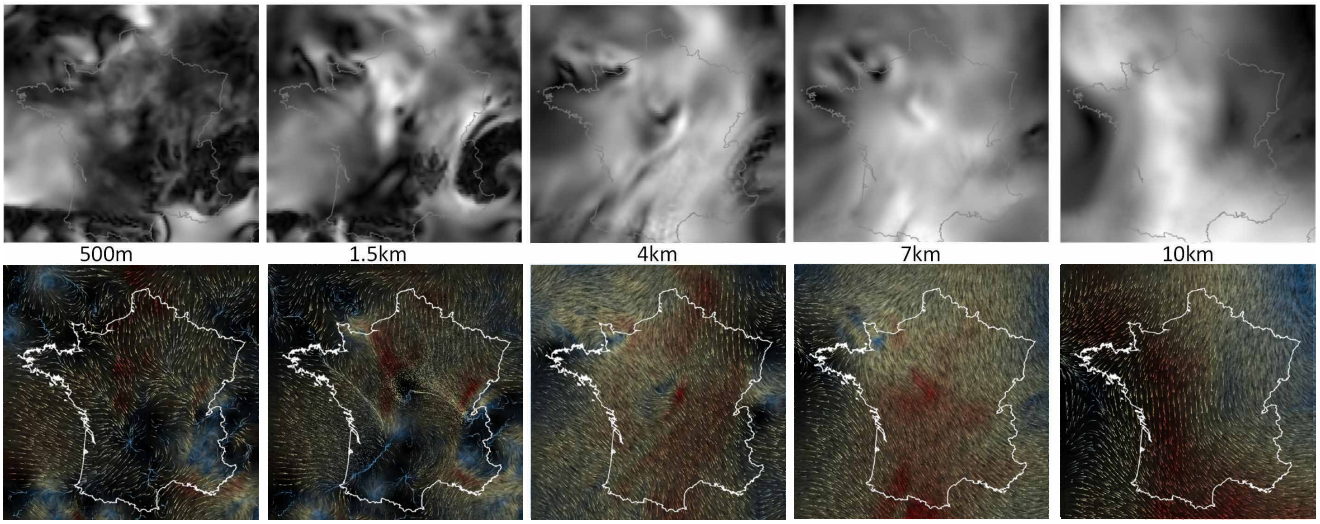

Figure 13: *Wind flow at different altitudes at 6 AM. Top: flow magnitude (grayscale colormap). Bottom: Our composite visualization encoding flow direction and magnitude. While this small-multiple visualization shows how wind evolves over altitude, one still needs some effort to understand such dynamics.*

obtained for such smaller kernels look practically identical to the one in Fig. 12. Hence, seeing this type of pattern in the direction field is a strong signal of bundling convergence problems.

Concluding, visualizing the field θ is a good *predictor* of how bundling will work for a given kernel size h. Note that this is an *a priori* instrument: We can visualize θ starting at the first bundling iteration, and in real time, as bundling progresses. Once we see undesired patterns, like the ones outlined above, we know that (1) bundling will not progress as desired (so we can stop it right away) and (2) how to change h to fix this. This contrasts the standard *a posteriori* assessment currently used in all bundling literature, where users run the *entire* bundling process, and when results are not good, must (1) change parameters (but it is not clear in which way) and (2) re-run the entire bundling process (which takes time). This also explains why two of the three bundling images in Fig. 12 appear suboptimal: They are, indeed, poor bundling results, which we detect early on by our visualization of θ.

7. Discussion

We now discuss our user study and composite visualization proposal.

Composition and user study: The compositing technique is based on the results of the user study, that has a limited number of users (15), datasets (9), tasks (1), flow features (7) and techniques (3). The evaluation of the compositing design has the same limitations. Yet, the two parts of our contribution can be separated: One can *e.g.* study more tasks or more visualization techniques, such as flow transport assessment, or more visualization techniques. We kept our scope limited, as outlined above, to be able to provide a more detailed quantitative evaluation. The same holds for the use of color: In our two studies, we did not measure the effect of color has for the tested tasks (critical point detection). Doing so would have been

possible, but would have significantly complicated the study, *e.g.*, by needing to choose various types of colormaps, what quantities these encode, and which tasks the final visualization should now answer. Finding which perception effects has color addition to the composite visualization is, however, an important question, which can be solved by a separate study.

Based on the obtained results, our compositing method (Sec. 4) can be *directly* used to design new visualization methods and/or mappings from feature types to optimal visualization methods: *Any* visualization method can be directly used by considering its output image. Furthermore, other parameters could be considered for the compositing, and different techniques assignments. Finally, even if the critical points are not optimally found, the distance-based blending ensures that the result still is a meaningful visualization.

Concerning the limited number of participants, one could have considered using *e.g.* crowdsourcing to improve it. We chose to use a controlled study to be able to control the experiment setup, something much harder to do with crowdsourcing [GMN*17]. For instance, having control over the study allowed us to ensure all participants did indeed follow the training and instructions, had access to identical computers for the experiments, and identical room conditions. Also, note that the participants in the first and second user studies were not the same. Furthermore, we wanted to keep the between-subject structure in the second user study, to make the comparison between the techniques possible, given also our number of participants.

Flow features and weighting: We targeted only basic flow features (7 types of critical points), for simplicity of implementation and presentation. Yet, *any* other flow feature detection method [PVH*03, GLL91] that outputs a *location* $\mathbf{x}$ can be directly used in Eqn. 4. If the method outputs a *scalar field* w, *e.g.*, the divergence of the input field $\mathbf{v}$, we can simply use w as weight map instead of the Shepard weights w_i (Sec. 4.1). For example, we could consider Helmholtz-Hodge decompositions [BNaPTB13] that separate any vector field into divergence-free, curl-free, and laminar components, or the algebraic multigrid based vector field decomposition in [GRP*04]. The magnitude of these components could be directly used as weight maps w_i in Eqn. 4, since they satisfy the partition-of-unity property required by weights. This may create better weight maps than the isotropic (Shepard-based) ones we currently use, having weights better aligned with the flow features. However, exploring this design space is not trivial, given the many methods (and their parameters) available for flow decomposition. We leave this topic as an important extension for future work.

Use cases: We showed the usefulness of our compositing technique to better understand wind and "bundling" flow map. Our technique usage is not limited to these specific flow data-sets and is applicable to any flow data with other application areas. Regarding our information visualization use case, we show how we can use our visualization technique to improve algorithm transparency, with a better understanding of complex parameters, and thus opens up some interesting direction for future work. For example, other algorithms, such as Optical Flow [HS81], also rely on vector field calculations, and could benefit from flow visualizations to produce a more transparent algorithm output.

Ease of use: Our compositing technique (Sec. 4) works automatically (no user parameter settings) for any 2D (time-dependent) vector field, and achieves 60 frames per second for fields up to 1024^2 data points, due to the high scalability of the considered visualizations (IBFV, OLIC, PS) and the critical point detection.

8. Conclusion

We have presented a two-part approach for visualizing vector fields using a data-driven combination of visualization methods. First, we performed a study that maps three mainstream visualization methods to types of vector field features that these can best assist in finding. Using the obtained insights, we propose a method to compose visualization methods locally, based on detected field patterns and optimal methods for these patterns. A second user study shows that this technique is faster and less error-prone than the original methods for critical-point finding and classification tasks. Finally, we demonstrate our method with two real-world datasets from both scientific and information visualization.

We next aim to study additional visualization methods, tasks, and flow field patterns to enrich the optimal mapping scheme from the last to the first. The extension of our approach to 3D vector fields is technically trivial, and to be pursued. Ultimately, we hope to create a framework that significantly simplifies the creation of effective vector field visualizations for a wide range or users, tasks, and problems.

9. Acknowledgment

This research was partly supported by SESAR H2020 project ENVISION grant agreement No 783270.

References

[BBB*19] BLASCHECK T., BESANÇON L., BEZERIANOS A., LEE B., ISENBERG P.: Glanceable visualization: Studies of data comparison performance on smartwatches. *IEEE TVCG 25*, 1 (2019), 630–640. 4

[BCP*12] BRAMBILLA A., CARNECKY R., PEIKERT R., VIOLA I., HAUSER H.: Illustrative flow visualization: State of the art, trends and challenges. In *Proc. Eurographics – State of the Art Reports* (2012). 1, 2

[BD17] BESANÇON L., DRAGICEVIC P.: The Significant Difference between p-values and Confidence Intervals. In *29ème conférence francophone sur l'Interaction Homme-Machine* (2017), AFIHM, (Ed.). 4

[BNaPTB13] BHATIA H., NORGARD G., AMD PEER-TIMO BREMER V. P.: The Helmholtz-Hodge decomposition – a survey. *IEEE TVCG 19*, 8 (2013), 1386–1404. 11

[BR98] BECKER J., RUMPF M.: Visualization of time-dependent velocity fields by texture transport. In *Proc. ViSC* (1998), pp. 91–101. 2

[Bun89] BUNING P.: Numerical algorithms in CFD post-processing. In *Computer Graphics and Flow Visualization in Computational Fluid Dynamics* (1989), Von Kármán Institute for Fluid Dynamics. 3

[BW08] BACHTHALER S., WEISKOPF D.: Animation of orthogonal texture patterns for vector field visualization. *IEEE TVCG 14*, 4 (2008), 741–755. 2

[CL93] CABRAL B., LEEDOM L.: Imaging vector fields using line integral convolution. In *Proc. ACM SIGGRAPH* (1993), pp. 263–270. 2

[Cum14] CUMMING G.: The new statistics: Why and how. *Psychological Science 25*, 1 (2014), 7–29. 4

[Dra16] DRAGICEVIC P.: Fair statistical communication in hci. In *Modern Statistical Methods for HCI* (2016), Springer, pp. 291–330. 4

[EHP*11] ERSOY O., HURTER C., PAULOVICH F., CANTAREIRO G., TELEA A.: Skeleton-based edge bundling for graph visualization. *IEEE TVCG 17*, 12 (2011), 2364–2373. 9

[ELPH19] ENGELKE W., LAWONN K., PREIM B., HOTZ I.: Autonomous particles for interactive flow visualization. *CGF 38*, 1 (2019), 248–259. 2

[GLL91] GLOBUS A., LEVIT C., LASINSKI T.: A tool for visualizing the topology of three-dimensional vector fields. In *Proceeding Visualization '91* (Oct 1991), pp. 33–40. doi:10.1109/VISUAL.1991.175773. 11

[GMN*17] GADIRAJU U., MOLLER S., NOLLENBURG M., SAUPE D., EGGER S., ARCHAMBAULT D., FISHER B.: Crowdsourcing versus the laboratory: towards human-centered experiments using the crowd. In *Evaluation in the Crowd. Crowdsourcing and Human-Centered Experiments* (2017), Springer, pp. 6–26. 11

[GRP*04] GRIEBEL M., RUMPF M., PREUSSER T., SCHWEITZER M. A., TELEA A.: Flow field clustering via algebraic multigrid. In *Proc. IEEE Visualization* (2004). 11

[HAG*14] HURTER C., ALLIGIER R., GIANAZZA D., PUECHMOREL S., ANDRIENKO G., ANDRIENKO N.: Wind parameters extraction from aircraft trajectories. *Computers, Environment and Urban Systems 47* (2014), 28 – 43. 8

[HET12] HURTER C., ERSOY O., TELEA A.: Graph Bundling by Kernel Density Estimation. *CGF 31*, 3p (June 2012), 865–874. 9

[HET13] HURTER C., ERSOY O., TELEA A.: Smooth bundling of large streaming and sequence graphs. In *Proc. IEEE PacificVis* (2013), pp. 201–210. 7

[HH89] HELMAN J., HESSELINK L.: Representation and display of vector field topology in fluid flow data sets. *IEEE Computer 22*, 08 (1989), 27–36. 2, 5

[HS81] HORN B. K., SCHUNCK B. G.: Determining optical flow. *Artificial Intelligence 17*, 1 (1981), 185 – 203. 11

[HTCT14] HURTER C., TAYLOR R., CARPENDALE S., TELEA A.: Color tunneling: Interactive exploration and selection in volumetric datasets. In *Proc. IEEE PacificVis* (2014), pp. 32–41. 7

[Hur15] HURTER C.: *Image-Based Visualization: Interactive Multidimensional Data Exploration*. Synthesis Lectures on Visualization. Morgan & Claypool, 2015. 9

[HYL*15] HO H.-Y., YEH I.-C., LAI Y.-C., LIN W.-C., CHERNG F.-Y.: Evaluating 2d flow visualization using eye tracking. *CGF 34*, 3 (2015), 501–510. 2

[JEH00] JOBARD B., ERLBACHER G., HUSSAINI Y.: Hardware-accelerated texture advection for unsteady flow visualization. In *Proc. IEEE Visualization* (2000). 2

[JEH01] JOBARD B., ERLBACHER G., HUSSAINI Y.: Lagrangian-eulerian advection for unsteady flow visualization. In *Proc. IEEE Visualization* (2001). 2

[JL97] JOBARD B., LEFER W.: The motion map: Efficient computation of steady flow animations. In *Proc. IEEE Visualization* (1997), pp. 323–328. 2

[JL00] JOBARD B., LEFER W.: Unsteady flow visualization by animating evenly-spaced streamlines. *CGF 19*, 3 (2000), 31–39. 2

[JWC*11] JÄNICKE H., WEIDNER T., CHUNG D., LARAMEE R. S., TOWNSEND P., CHEN M.: Visual reconstructability as a quality metric for flow visualization. *CGF 30*, 3 (2011), 781–790. 2

[KKKW05] KRUGER J., KIPFER P., KONCLRATIEVA P., WESTERMANN R.: A particle system for interactive visualization of 3D flows. *IEEE TVCG 11*, 6 (2005), 744–756. 3

[KML99] KIRBY R. M., MARMANIS H., LAIDLAW D. H.: Visualizing multivalued data from 2d incompressible flows using concepts from painting. In *Proc. IEEE Visualization* (1999), pp. 333–340. 2

[KSW*12] KARCH G. K., SADLO F., WEISKOPF D., MUNZ C.-D., ERTL T.: Visualization of advection-diffusion in unsteady fluid flow. *Comp Graph Forum 31*, 3 (2012), 1105–1114. 2

[LCE*12] LIU Z., CAI S., EDWARD SWAN, J., MOORHEAD, R. J., MARTIN J. P., JANKUN-KELLY T. J.: A 2D flow visualization user study using explicit flow synthesis and implicit task design. *IEEE TVCG 18*, 5 (2012), 783–796. 2, 3, 4

[LHD*04] LARAMEE R. S., HAUSER H., DOLEISCH H., VROLIJK B., POST F. H., WEISKOPF D.: The state of the art in flow visualization: Dense and texture-based techniques. *CGF 23*, 2 (2004), 203–221. 1

[LHT17] LHUILLIER A., HURTER C., TELEA A.: State of the art in edge and trail bundling techniques. *CGF* (2017). 9

[LHZP07] LARAMEE R. S., HAUSER H., ZHAO L., POST F. H.: Topology-based flow visualization, the state of the art. In *Topology-based Methods in Visualization* (2007), Hauser H., Hagen H., Theisel H., (Eds.), Springer, pp. 1–19. 1, 2, 5

[LJH03] LARAMEE R. S., JOBARD B., HAUSER H.: Image space based visualization of unsteady flow on surfaces. In *Proc. IEEE Visualization* (2003). 2

[LJL04] LEFER W., JOBARD B., LEDUC C.: High-quality animation of 2D steady vector fields. *IEEE TVCG 10*, 1 (2004), 2–14. 2

[LKJ*05] LAIDLAW D. H., KIRBY R. M., JACKSON C. D., DAVIDSON J. S., MILLER T. S., DA SILVA M., WARREN W. H., TARR M. J.: Comparing 2D vector field visualization methods: A user study. *IEEE TVCG 11*, 1 (2005), 59–70. 1, 2, 3

[LTH08] LI G. S., TRICOCHE X., HANSEN C. D.: Physically-based dye advection for flow visualization. *Comp Graph Forum 27*, 3 (2008), 727–734. 2

[LWSH04] LARAMEE R. S., WEISKOPF D., SCHNEIDER J., HAUSER H.: Investigating swirl and tumble flow with a comparison of visualization techniques. In *Proc. IEEE Visualization* (2004), pp. 322–330. 1

[MB95] MAX N., BECKER B.: Flow visualization using moving textures. *Visualizing time varying data, Williamsburg, VA (United States)* (4 1995). 2

[MK13] MATVIENKO V., KRÜGER J.: A metric for the evaluation of dense vector field visualizations. *IEEE TVCGs 19*, 7 (2013), 1122–1132. 2

[MLP*10] MCLOUGHLIN T., LARAMEE R. S., PEIKERT R., POST F. H., CHEN M.: Over two decades of integration-based, geometric flow visualization. *CGF 29*, 6 (2010), 1807–1829. 1, 2

[MSIMI*08] MARTIN J. P., SWAN II J. E., MOORHEAD II R. J., LIU Z., CAI S.: Results of a user study on 2D hurricane visualization. *CGF 27*, 3 (2008), 991–998. 8

[PAPB19] PEÑA-ARAYA V., PIETRIGA E., BEZERIANOS A.: A comparison of visualizations for identifying correlation over space and time. *IEEE transactions on visualization and computer graphics 26*, 1 (2019), 375–385. 4

[PHT15] PEYSAKHOVICH V., HURTER C., TELEA A.: Attribute-driven edge bundling for general graphs with applications in trail analysis. In *Proc. IEEE PacificVis* (2015), pp. 39–46. 9

[PVH*03] POST F. H., VROLIJK B., HAUSER H., LARAMEE R. S., DOLEISCH H.: The state of the art in flow visualisation: Feature extraction and tracking. *CGF 22*, 4 (2003), 775–792. 1, 2, 11

[PW08] PINEO D., WARE C.: Neural modeling of flow rendering effectiveness. In *Proc. ACM APGV* (2008), pp. 171–178. 2

[PW13] PILAR D. H. F., WARE C.: Representing flow patterns by using streamlines with glyphs. *IEEE TVCG 19*, 8 (2013), 1331–1341. 2

[RGG19] RIMENSBERGER N., GROSS M. H., GÜNTHER T.: Visualization of clouds and atmospheric air flows. *IEEE CG & A 39*, 1 (2019), 12–25. 8

[SFL*04] SOBEL J. S., FORSBERG A. S., LAIDLAW D. H., ZELEZNIK R. C., KEEFE D. F., PIVKIN I., KARNIADAKIS G. E., RICHARDSON P., SWARTZ S.: Particle flurries. *IEEE CG & A 24*, 2 (2004), 76–85. 2

[She68] SHEPARD D.: A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference* (New York, NY, USA, 1968), ACM '68, ACM, pp. 517–524. 5

[SK16] SCHROEDER D., KEEFE D.: Visualization-by-sketching: An artist's interface for creating multivariate time-varying data visualizations. *IEEE TVCG 22*, 1 (2016), 877–885. 1, 2

[SLWS08] SALZBRUNN T., LEITTE H., WISCHGOLL T., SCHEUERMANN G.: The state of the art in flow visualization: Partition-based techniques. In *Proc. SimVis* (2008). 1, 2

[SS89] SETHIAN J. A., SALEM J. B.: Animation of interactive fluid flow visualization tools on a data parallel machine. *Intl J High Performance Comput Appl 3*, 2 (1989), 10–39. 1

[SS07] SALZBRUNN T., SCHEUERMANN G.: Streamline predicates as flow topology generalization. In *Topology-based Methods in Visualization* (2007), Springer, pp. 65–77. 2

[Sun03] SUNDQUIST A.: Dynamic line integral convolution for visualizing streamline evolution. *IEEE TVCG 9*, 3 (2003), 273–282. 2, 3

[The19] THE AUTHORS: User evaluation data logs and scripts, 2019. https://mjlobo.bitbucket.io/suppmaterial_eurovis/indexsupp.html. 4

[Tru84] TRUESDELL C.: *Daniel Bernoulli's Hydrodynamica (1960)*. Springer New York, New York, NY, 1984, pp. 209–211. 8

[TSH01] TRICOCHE X., SCHEUERMANN G., HAGEN H.: Topology-based visualization of time-dependent 2D vector fields. In *Proc. Data Visualization* (2001), pp. 117–126. 2

[TvW99] TELEA A., VAN WIJK J. J.: Simplified representation of vector fields. In *Proc. IEEE Visualization* (1999), pp. 56–44. 2

[TvW03] TELEA A., VAN WIJK J. J.: 3D IBFV: Hardware-accelerated 3D flow visualization. In *Proc. IEEE Visualization* (2003), pp. 131–139. 1, 2

[UIL*06] URNESS T., INTERRANTE V., LONGMIRE E., MARUSIC I., O'NEILL S., JONES T. W.: Strategies for the visualization of multiple 2d vector fields. *IEEE CG & A 26*, 4 (2006), 74–82. 2

[vdZCT16] VAN DER ZWAN M., CODREANU V., TELEA A.: CUBu: Universal real-time bundling for large graphs. *IEEE TVCG 22*, 12 (2016), 2550–2563. 9

[VP04] VERMA V., PANG A.: Comparative flow visualization. *IEEE TVCG 10*, 6 (2004), 609 – 624. 2

[vW02] VAN WIJK J. J.: Image based flow visualization. *ACM Trans. Graph. 21*, 3 (July 2002), 745–754. 2, 3, 7

[vW03] VAN WIJK J. J.: Image based flow visualization for curved surfaces. In *Proc. IEEE Visualization* (2003), pp. 166–174. 2

[WBM*16] WARE C., BOLAN D., MILLER R., ROGERS D. H., AHRENS J. P.: Animated versus static views of steady flow patterns. In *Proc. ACM Symp. on Applied Perception (SAP)* (2016), pp. 77–84. 2

[WG97] WEGENKITTL R., GRÖLLER E.: Fast oriented line integral convolution for vector field visualization via the internet. In *Proc. IEEE Visualization* (1997), pp. 309–316. 3

[WGP97] WEGENKITTL R., GROLLER E., PURGATHOFER W.: Animating flow fields: Rendering of oriented line integral convolution. In *Proc. IEEE Computer Animation* (1997), pp. 15–24. 2, 3

[WP13] WARE C., PLUMLEE M. D.: Designing a better weather display. *Information Visualization 12*, 3-4 (2013), 221–239. 2, 3, 8

[XLS10] XU L., LEE T., SHEN H.: An information-theoretic framework for flow visualization. *IEEE TVCG 16*, 6 (2010), 1216–1224. 2