SurfRiver: Flattening Stream Surfaces for Comparative Visualization

Jun Zhang, Jun Tao, *Member, IEEE*, Jian-Xun Wang, and Chaoli Wang, *Senior Member, IEEE*

Abstract—We present SurfRiver, a new visual transformation approach that flattens stream surfaces in 3D to rivers in 2D for comparative visualization. Leveraging the TextFlow-like visual metaphor, SurfRiver untangles the convoluted individual stream surfaces along the flow direction and maps them along the horizontal direction of the abstract river view. It stacks multiple surfaces along the vertical direction of the river view. This visual mapping makes it easy for users to track along the flow direction and align stream surfaces for comparative study. Through brushing and linking, the river view is connected to the spatial surface view for collective reasoning. SurfRiver can be used to examine a single stream surface, investigate seeding sensitivity or variability of a family of surfaces from a group of related seeding curves, or explore a collection of representative surfaces. We describe our optimization solution to achieve the desirable mapping, present SurfRiver interface and interactions, and report results from different flow fields to demonstrate its efficacy. Feedback from a domain expert also indicates the promise of SurfRiver.

Index Terms—Flow visualization, stream surfaces, visual transformation, comparative visualization.

1 Introduction

Fluid simulation and flow visualization are essential in many dynamic systems that dominate various physical and natural phenomena. Among the popular integration-based visualization techniques, line-based techniques have made significant advances over the years, providing a sharp contrast to surface-based techniques. To construct flow surfaces, we place particles on a *seeding rake* or *seeding curve* in the given vector field and advect them over time. The collective traces that the particles follow yield *stream surfaces* for steady flow and *path surfaces* for unsteady flow, depicting the *folding*, *shearing*, and *twisting* behaviors of the underlying flow. These surfaces enhance the visual perception of convoluted flow structures and facilitate an intuitive understanding of flow patterns, providing illustrative capabilities and improved visualization over simple integral curves [21]. We refer interested readers to a survey of surface-based flow visualization [11] for an overview.

Existing methods for surface-based flow visualization still face three critical challenges in surface generation, visualization, and analytics. First, surface generation requires either the knowledge to place seeds for creating informative surfaces (*surface placement*) or the solution to choose characteristic surfaces from a pool of densely-traced ones (*surface selection*). Second, unlike flow lines, flow surfaces are more likely to create visual occlusion and clutter. This problem may be due to multiple surfaces that occlude one another, a single surface that produces strong self-occlusion, or a combination of both [11]. This challenge must be addressed so that insightful visualizations can be generated. Third, prior works on flow surface visualization are mostly automatic. Although the automated approach has its benefits in many scenarios, users are often given standard interaction options, which offers little opportunity to flexibly query the visualizations or freely explore the results.

In this paper, we advocate a visual transformation approach for the comparative study of stream surfaces. We introduce SurfRiver that addresses the need for visual exploration of a single surface and comparative visualization of a family of surfaces by transforming them into the TextFlow-like visual representation [10]. By flattening and aligning *surfaces* in 3D to *rivers* in 2D, it allows clear examination and comparison of multiple surfaces. SurfRiver consists of two views, *spatial surface view* and *abstract river view*. The spatial surface view shows the stream surfaces in the original 3D space. The abstract river view displays the flattened surfaces as rivers in the 2D space. The two views are dynamically linked via brushing and linking to enables users to gain a comprehensive examination of surfaces from different perspectives. In terms of interaction, SurfRiver supports the exploration of the river view via different mapping schemes and the alignment of surfaces for better observation of their 3D spatial relationships.

The contribution of this work is as follows. First, we design a novel visual transformation solution that flattens stream surfaces to support clear visualization and analytical exploration. Second, we propose an optimization scheme and visual mappings that build the intuition between the spatial surface view and the abstract river view. Third, we develop a suite of interactions to enable the exploratory and comparative study of single and multiple surfaces. Fourth, we present a list of cases to demonstrate our solution's efficacy and solicit feedback from a domain expert.

- *J. Zhang is with the Department of Computer Science and Engineering, University of Notre Dame and the School of Computer Science and Engineering, Sun Yat-sen University, E-mail: jzhang28@nd.edu.*
- *J. Tao is with the School of Computer Science and Engineering, Sun Yat-sen University and the National Supercomputer Center in Guangzhou, China. E-mail: taoj23@mail.sysu.edu.cn. He is the corresponding author.*
- *J.-X. Wang is with the Department of Aerospace and Mechanical Engineering, University of Notre Dame, E-mail: jwang33@nd.edu.*
- *C. Wang is with the Department of Computer Science and Engineering, University of Notre Dame, E-mail: chaoli.wang@nd.edu.*

2 Related Work

In the context of flow visualization, we review related work on surface placement and selection, surface construction, surface rendering, and comparative visualization.

Surface placement and selection. An early work presented by van Wijk [35] generates stream surfaces *implicitly* where continuous scalar values are specified for boundary voxels and scalar values for interior voxels are computed through backward streamline tracing. Stream surfaces are then constructed through isosurface extraction in the scalar field. Following this implicit approach, Cai and Heng [8] constructed the *principal stream function* based on normal vectors to the principal stream surface at velocity points. The principal stream function is a scalar field that describes the velocity direction of the flow field and can be visualized through volume rendering. Edmunds et al. [12] performed hierarchical clustering of local flow properties to locate seeding positions associated with important flow structures, from which seeding curves are generated through the *curvature field* for stream surface propagation. Martinez Esturo et al. [27] favored surfaces where the flow is aligned with *principal curvature directions*. Simulated annealing was used to determine a stream surface that is globally optimal in terms of quality measures. Schulze et al. [32] selected a set of globally-optimal, mutually-distant stream surfaces that optimizes global stream surface quality measures. Other works find surfaces that meet specific desired properties best, for example, aligning with the flow or orthogonal to it [13], minimizing stretch [3], or seeding along selected tensor lines of the similarity tensor field [7]. Tao and Wang [33] advocated a *sketch-based* approach that allows users to iteratively draw strokes directly on densely-seeded streamlines to identify suitable seeding curves and guide stream surface placement. Han et al. [19] leveraged an autoencoder to learn *latent* stream surface features and projected these feature descriptors into a 2D space for interactive surface clustering and customized representative selection. In this work, instead of directly tackling surface placement or selection, we study seeding sensitivity or variability of a family of stream surfaces produced from a group of related seeding curves. Besides, we also consider a set of representative surfaces in the investigation.

Surface construction. Hultquist [20] was the first to present the solution to stream surface construction based on the *front-advancing* approach. His algorithm advances a seeding rake (i.e., straight seeding curve) as the front to generate stream ribbons from adjacent streamline pairs. Scheuermann et al. [30] presented a *tetrahedra-based* stream surface algorithm that utilizes an analytic flow solution for linear interpolation over tetrahedral grids. Their approach can automatically adapt to the grid resolution. Garth et al. [16] employed an *arc-length based* solution to streamline integration and considered additional criteria such as surface curvature and singularities for front refinement. Their solution can handle flows of inhomogeneous magnitude that are not well handled by Hultquist's algorithm [20]. Garth et al. [15] advocated a two-step approach for surface generation. The two steps separate *surface approximation* (which generates a skeleton of the integral surface) from *surface representation* (which generates a well-conditioned triangulation). Other surface construction techniques include *point-based* [29] and *quad-based* [28] approaches. Based on a scaled version of the flow field, Schulze et al. [31] designed an integrator that enforces the flow-orthogonal front line to generate well-behaved quad-dominant meshes. In this work, we follow the easy integral surface construction method proposed by McLoughlin et al. [28] to generate stream surfaces from steady vector fields.

Surface rendering. Flow surfaces can be rendered directly as mesh geometry with lighting and color mapping. They can also overlay with glyphs (e.g., arrows), integral lines (e.g., streamlines), or textures (e.g., LIC textures) to improve the perception of surface flows. Introducing illustrative techniques to surface rendering can reduce visual occlusion and clutter, and enhance the depth and spatial perception of flow features and structures. Born et al. [5] used illustrative surface streamlines to highlight flow directions and singularities while *contour lines* and *half-toning* were employed to depict the shape of stream surfaces. Hummel et al. [21] considered transparency and texturing to enhance the shape and directional information with *screen-space curvature approximation*. Carnecky et al. [9] enabled nonlocal transparency enhancement and achieved expressive surface rendering using the *illustration buffer*. Günther et al. [17] studied *opacity optimization* for surfaces and further extended this solution to handle all geometry types (points, lines, and surfaces) in a single framework [18].

Comparative visualization. Researchers have investigated transforming flow lines and surfaces for comparative visualization and visual reasoning. Verma and Pang [36] argued the need for comparative flow visualization, classified three levels (*image-level*, *data-level*, and *feature-level*) of comparison, and presented their solutions on streamline and stream ribbon comparison. *Curved planar reformation* (CPR) [22], [23] and *curved surface reformation* (CSR) [2] were used to visualize tubular structures such as blood vessels. Similar works have been done to enable comparative visualization of 1D flow lines via volume reformation [24] and line straightening [1]. More abstract representation changes transform flow lines, features, and space-time regions into a tree or graph representation. Such examples include HemoVis [4] for representing coronary artery trees as well as FlowGraph [25], [26] and semantic flow graph [34] for exploring general fluid flows. However, comparative 2D surface visualization has not been well investigated in this context. Brambilla et al. [6] reformatted time surfaces into a planar space to enable *juxtaposition*, *superimposition*, and *side-by-side* comparative visualization. The third dimension is used to stack formatted surfaces. In contrast, using a river metaphor, we flatten stream surfaces into rivers where the flow direction always follows the *horizontal* direction, and multiple surfaces are stacked along the *vertical* direction for comparative visualization.

3 SurfRiver Overview

The rationale to design SurfRiver stems from highly-sensitive surface seeding in real-world flow fields. For line seeding, small perturbations in the seeding location could lead to dramatically different streamlines. Surface seeding has a much higher degree of freedom to vary as the variation in location, length, and shape of seeding curves would

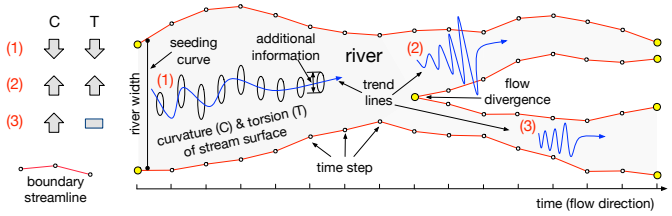


Fig. 1: Illustration of the mapping of flow features of a stream surface to visual attributes of a river in SurfRiver. The trend lines indicate that along the flow direction, both curvature and torsion values decrease for (1), both values increase for (2), and the curvature value increases while the torsion value remains the same for (3).

lead to different stream surfaces. Our primary goal is to enable visual examination and exploration of such a *family* of stream surfaces to study seeding sensitivity or variability. Besides mapping those stream surfaces that are more likely to be similar to each other, we also explore the mapping of a set of *representative* surfaces, which can be obtained from existing works [19], [32].

As sketched in Figure 1, similar to TextFlow [10], where the horizontal direction represents time, that direction of SurfRiver represents the flow direction (i.e., the integration time for steady flow). Each *river* in SurfRiver represents a stream surface, and the river always flows from left to right. Along the vertical direction, a river widens or narrows in accordance with the timelines (i.e., advancing fronts) to depict the varying widths of the surface at different time steps. We map the boundary streamlines to the river's boundary, and the average *curvature* and *torsion* values of the timelines to a *trend line* at the center of each river branch indicated by the *amplitude* and *frequency* of the wave pattern, respectively. Finally, we optionally display ellipses along the trend line to show additional flow information.

This added TextFlow-like view brings a clear advantage: we essentially "flatten" 3D stream surfaces to 2D rivers, which reduce or eliminate occlusion. SurfRiver allows users to examine a family of surfaces and investigate *when*, *where*, and *how* they differ from each other. To align multiple rivers in SurfRiver, we take their seeding curves as the reference, so that stream surfaces are aligned according to their spatiotemporal arrangement. We use a linked spatial view to examine the surface family in 3D.

4 SurfRiver Layout

SurfRiver aims at unfolding 3D stream surfaces on a 2D plane. The horizontal axis of SurfRiver preserves the temporal order of the timelines and aligns the timelines according to their similarities, and the vertical axis preserves the spatial distance between aligned timelines. We achieve this goal by formulating the creation of SurfRiver as a graph layout problem. Each *node* in the graph represents a *timeline* of a stream surface, and each *edge* enforces a *constraint* between two timelines, which will be elaborated in this section. Unlike the conventional 2D graph layout algorithm, we compute each node's x-coordinate and y-coordinate separately in two stages: *horizontal alignment* and *vertical ordering*. For each stage, we solve the respective coordinates by minimizing an energy function. Once the coordinates are determined, we extend the node from a point to a line segment and connect the line segments' endpoints to form the river's boundary.

At the first stage, horizontal alignment "warps" the surfaces non-linearly along the x-axis, so that the nodes corresponding to similar timelines have similar x-coordinates (the layout along the x-axis is a compromise between temporal alignment and similarity alignment). The similarity of two timelines may be given by their spatial proximity or similarity of other features (e.g., average curvature and torsion values). Horizontal alignment builds the correspondence between timelines on different surfaces along the x-axis. At the second stage, vertical ordering places the nodes along the y-axis to resemble the spatial relationships between aligned timelines. Two timelines are considered to be aligned if their x-coordinates computed at the first stage are similar. Additional constraints are considered to maintain a smooth flow outline and reduce visual clutter.

Figure 2 gives an overview of the input timelines to each energy term that we will explain next for horizontal alignment and vertical ordering. In the following, we use l_i to denote the i-th timeline, and $t(l_i)$ to denote a time step of l_i with respect to its corresponding seeding curve, a.k.a. the initial timeline ($t(l_i) = 0$ if l_i is a seeding curve). We further use x_i and y_i to denote the x- and y-coordinates of the node corresponding to l_i, respectively. For simplicity, we will refer to x_i/y_i as "the x/y-coordinate of timeline l_i".

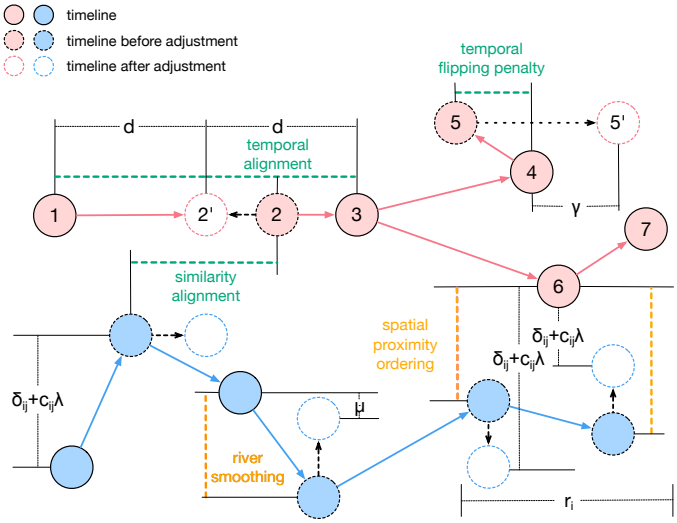


Fig. 2: Illustration of the input for each energy term in horizontal alignment and vertical ordering.

4.1 Horizontal Alignment

For horizontal alignment, we consider three energy terms: *similarity alignment*, *temporal alignment*, and *temporal flipping penalty*. Similarity alignment aims to achieve horizontal proximity for similar timelines. Temporal alignment maintains the order of timelines on the same surface. Temporal flipping penalty is applied when the order of timelines is violated.

Similarity alignment. The similarity alignment energy term E_s horizontally pulls nodes on the same surface closer if their corresponding timelines are similar. Specifically, this term enforces a timeline to share the same x-coordinate

in SurfRiver with its most similar timeline, as shown in Figure 2. Formally, this can be expressed as

$$E_s = \sum_{l_i \in S, l_j \notin S} w_s (x_i - x_j)^2, \text{ where } j = \underset{k}{\operatorname{argmin}}\ \delta_{ik}, \quad (1)$$

where S denotes a stream surface, w_s is the weight for similarity assignment, and δ_{ik} is the distance between timelines l_i and l_k. If the spatial proximity is considered, we use the *mean of closest point distances* (MCPD) between l_i and l_k. If the similarity of features is considered, we use the *Jensen-Shannon divergence* (JSD) between the curvature and torsion distributions of l_i and l_k.

Temporal alignment. As shown in Figure 2, the temporal alignment energy term E_t is used to enforce a constant horizontal gap between two neighboring timelines on the same surface S

$$E_t = \sum_{l_i, l_j \in S} (1 - w_s) \left((x_i - x_j) - d \right)^2, \text{ where } t(l_i) - t(l_j) = 1, \quad (2)$$

where d is a constant separation distance. We set $d = 4.0$ for all data sets we experimented with. Note that we use $1 - w_s$ as the weight of this term, so that adjusting w_s leads to a desired balance between similarity alignment and temporal alignment.

Temporal flipping penalty. Temporal flipping refers to an undesired layout where two timelines appear in reversed order along the x-axis, as shown in Figure 2. This often happens in a complex flow or around a vortex where two timelines are temporally distant but spatially close. In this case, we apply a large penalty to flip the timelines back to their correct temporal order

$$E_f = \sum_{l_i, l_j \in S} \alpha \left((x_i - x_j) - \gamma \right)^2, \quad \text{where } t(l_i) - t(l_j) = 1 \text{ and } x_i > x_j, \quad (3)$$

where α is a relatively large constant to enforce the constraint, and γ is a small constant, to simulate spatial closeness. In this paper, we set $\alpha = 1000$ and $\gamma = 0.1$ for all data sets we experimented with.

As shown in Figure 3 (a), surfaces A and B are traced from two very close and similar seeding curves shown on the left. As A goes through the obstacle located at the center of the domain, it starts twisting while B continues to flow more smoothly. With temporal alignment shown in Figure 3 (b), the blue timeline on A is aligned with the orange timeline on B, even though the orange timeline on B is spatially close to the black one on A (refer to Figure 3 (a)). By enabling similarity alignment with MCPD, we can align the orange and black timelines, as shown in Figure 3 (c).

4.2 Vertical Ordering

For vertical ordering, we consider two energy terms: *spatial proximity ordering* and *river smoothing*. Spatial proximity ordering preserves the proximity of aligned timelines along the vertical direction. River smoothing reduces the difference between neighboring timelines.

Spatial proximity ordering. As shown in Figure 2, the spatial proximity ordering energy term E_p enforces the

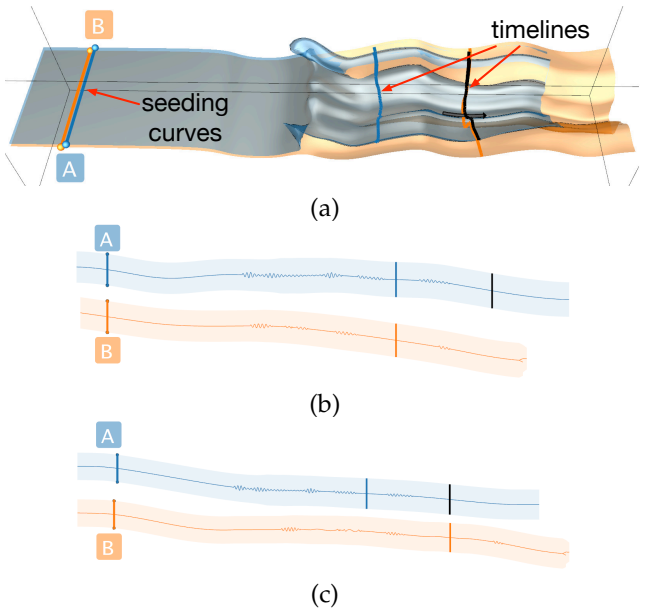


Fig. 3: Aligning two stream surfaces flowing through an obstacle of the square cylinder data set. (a) the surface view. (b) and (c) the corresponding river views with temporal alignment and similarity alignment, respectively.

y-coordinates of aligned timelines to reflect their original MCPD in the 3D space

$$E_p = \sum_{l_i, l_j \in \mathbf{S}} w_{ij} \left(|y_i - y_j| - (\delta_{ij} + c_{ij}\lambda) \right)^2, \text{ where}$$
$$w_{ij} = \frac{1}{\delta_{ij}^2} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{|x_i - x_j|}{2} \right) \text{ and } |x_i - x_j| \le r_i, \quad (4)$$

where $\mathbf{S}$ denotes the family of surfaces or the set of representative surfaces, $|y_i - y_j|$ is the absolute difference between y_i and y_j, δ_{ij} is the distance between timelines l_i and l_j given by their MCPD, and w_{ij} is a weight given by δ_{ij} and the horizontal distance $|x_i - x_j|$. λ is a constant gap to separate different river branches (due to flow divergence). λ is modulated by c_{ij}, which is the vertical ordering difference of two rivers when l_i and l_j are on different rivers, or a constant when l_i and l_j are on different branches of the same river. r_i is a cutoff radius to reduce the number of constraints. In this paper, we set $r_i = 0.2 \max_{k \in n} \delta_{ik}$, where $l_i \in S$, $l_k \in S'$, $S \in \mathbf{S}$, $S' \in \mathbf{S}$, and n is the number of timelines on S'. The first part of w_{ij} based on δ_{ij} is included in following stress majorization [14]. The second part of w_{ij} weights the influence of a timeline based on the horizontal distance using a Gaussian function. Intuitively, when $|x_i - x_j|$ between timelines l_i and l_j increases, their relative vertical order becomes less obvious. Therefore, we reduce the weight to loosen the constraint. λ is a parameter to enforce a constant vertical gap between river branches. We use $\lambda = 0$ when only the vertical order of timelines is considered. In other cases, users may specify λ for their desired visual effect. A small λ preserves the 3D spatial relationships in the 2D space, while a large λ increases gaps between river branches to reduce visual clutter.

Figure 4 shows an example with two different λ values. We can see that the river mapping can be adjusted to reduce or avoid branches crossing. When only the spatial proximity

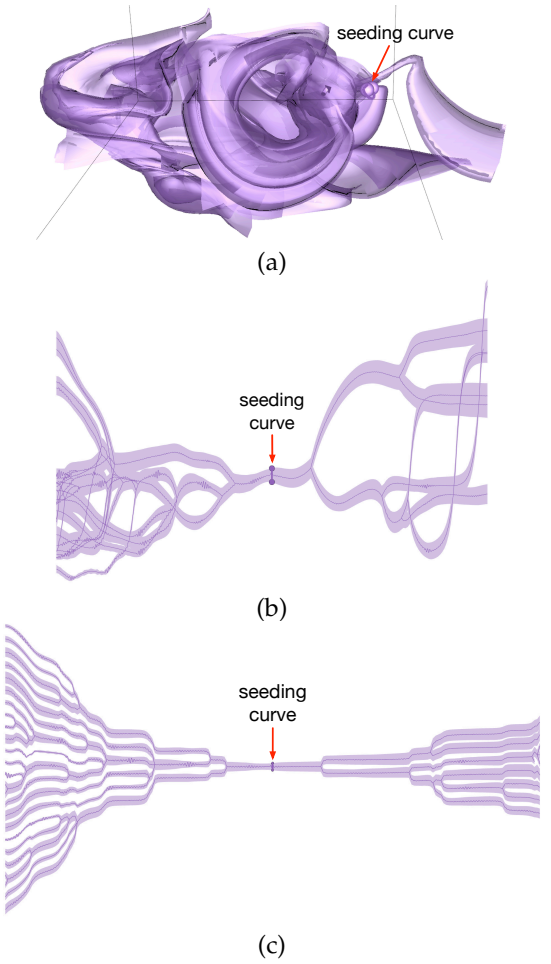


Fig. 4: The impact of λ using the crayfish data set showing a complex stream surface in (a) and the corresponding rivers with crossed and separated branches, respectively, in (b) and (c). For (b), $\lambda = 0$. For (c), $\lambda = 9$.

of timelines is considered, river branches tend to cross with each other, as shown in Figure 4 (b). Timelines on the same branch are prone to be pulled towards timelines residing in diverse regions, thus causing the entanglement. By setting λ to a large value, usually 100 times of the MCPD, we enforce timelines to maintain a constant distance between them on sibling branches, as shown in Figure 4 (c).

River smoothing. The river smoothing energy term E_r avoids sharp turns along a river and maintains smooth river boundaries for aesthetic purposes. It also helps enhance perception, as fluctuation in rivers may lead to difficulty in identification, which prevents users from tracking local features of the original surfaces. As shown in Figure 2, this term minimizes the vertical distance between neighboring timelines as

$$E_r = \sum_{l_i, l_j \in S} w_r \left(|y_i - y_j| - \mu\right)^2, \text{ where } t(l_i) - t(l_j) = 1, \tag{5}$$

where w_r is the weight for river smoothing, and μ is a very small constant to enforce vertical proximity. In this paper, we set $\mu = 0.00001$ for all data sets we experimented with.

As shown in Figure 5 (a), five stream surfaces are traced with their flow direction pointing from the domain's center to the boundary as indicated by the arrows. No divergence takes place in this case. The stack-like 3D spatial relationship is reflected in the 2D river view, as shown in Figure 5 (b), where these rivers are arranged from top to bottom. As the stream surfaces flow outward, they gradually separate. Therefore, the rivers' right ends slightly deviate from their left ends as they are pushed around by each other.

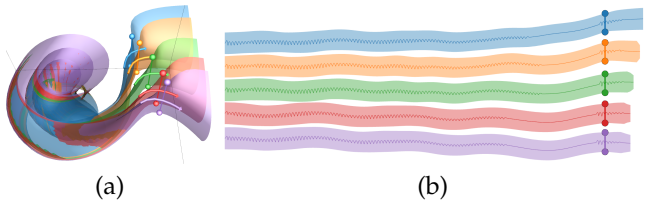


Fig. 5: Vertical ordering using the tornado data set. Spatial ordering of stream surfaces (a) is kept in the river view (b).

4.3 Energy Minimization

Horizontal alignment. The energy function of horizontal alignment can be minimized using a linear solver. The energy terms E_s, E_t, and E_f can be written as a series of linear equations, forming a sparse overdetermined linear system in the form of $\mathbf{Ax} = \mathbf{b}$. Although an exact solution usually does not exist for this kind of system, solving the linear system in the least square sense can minimize these energy terms. Specifically, a term in the form of $w((x_i - x_j) - d)$ is represented as a row in the matrix $\mathbf{A}$: the i-th element in this row is w, the j-th element is $-w$, and all other elements are zero. The corresponding element in $\mathbf{b}$ is wd.

Vertical ordering. The energy function of vertical ordering cannot be minimized using the linear solver due to the absolute difference between the unknown variables. Unlike horizontal alignment, where the order of timelines is enforced by their temporal relationships, the vertical order of timelines is unknown. Thus, a linear solver is unlikely to achieve an optimal vertical order of timelines without considering the absolute difference. Therefore, we use stress majorization [14], which minimizes loss functions in the form of $\sum w_{ij}(|y_i - y_j| - \delta_{ij})^2$. However, we find that stress majorization is often numerically unstable, as the 1D layout problem allows limited room for the nodes to switch their order. This problem may not be feasible to solve theoretically. In our experiment, we adopt the following empirical strategies to improve the solution's quality.

First, we solve the vertical ordering problem in two rounds. In the first round, we aim to solve a simplified energy function with $\lambda = 0$, which means that the constant separation gap is not considered. The second run uses the actual λ to separate the river branches. The y-coordinates are initialized by translating the branches from the first round's solution. Each branch is translated by a distance based on their vertical order. With the new initial values, we use stress majorization again to minimize the actual energy function. Second, we find that 1D stress majorization is likely to shift the positions uniformly without changing their relative distance. This issue may delay convergence and lead to numerical overflow. Therefore, we eliminate the shifting by deducting the average from the positions after every iteration of stress majorization. Third, with the

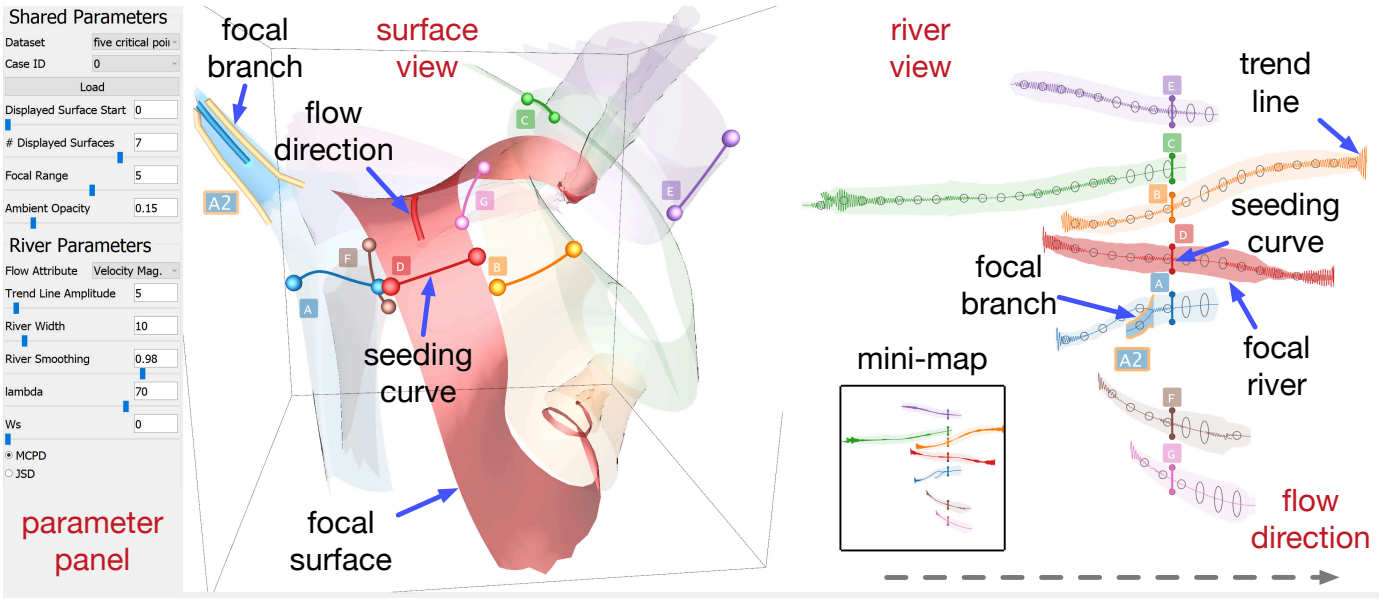


Fig. 6: SurfRiver consists of three parts: parameter panel, surface view, and river view. The parameter panel includes shared parameters for controlling parameters of both views and river parameters for adjusting desired river layouts. The surface view and river view are synchronized via brushing and linking to enable analytical reasoning.

vertical order decided, the signs of the absolute difference are known. The energy terms E_p and E_r can be converted to a linear system. We find that using the linear solver with the given signs often further minimizes the energy.

5 SurfRiver Interface and Interaction

As shown in Figure 6, the SurfRiver interface consists of two coordinated views: a *surface view* that renders the stream surfaces in the original 3D space, and a *river view* that displays the corresponding "flattened" rivers in the 2D layout. Users can scale, rotate, and translate the surface view, and scale and translate the river view. Both views are dynamically linked together via brushing and linking. The majority of interaction begins with the river view due to its 2D nature for easy navigation and convenient exploration.

5.1 Visual Mapping

For each stream surface in the surface view, a river is created in the river view. We visually pair the surface and its corresponding river by rendering them with a unique color. In addition, labels are added for clear correspondence. All seeding curves are displayed with two endpoints drawn for differentiation from other timelines. Ideally, the river should serve as a concise visual representation, allowing users to obtain the original surface's shape information and make mental connections between the river and the surface.

Trend line. To provide the shape information on a 2D river, we visually encode the average curvature and torsion values following the timelines as the trend line by deforming each river branch's center line into a sine wave. The amplitude of the sine wave represents the curvature, and the frequency represents the torsion. Intuitively, the laminar flow has smooth and flat curves, while the more complex flow exhibits larger amplitudes and higher frequencies. Figure 7 shows such an example where the segments at both ends of the trend line reflect the curly nature of the underlying stream surface. Furthermore, we optionally display ellipses along the trend line to show additional information. If this feature is turned on, we display velocity magnitude (by default) or vorticity magnitude. A constant sample rate for the ellipses is selected for each data set, depending on the number of timelines. The sample rate is usually between three and five (meaning that we display an ellipse for every three to five timelines along the trend line). In such an ellipse, the length of the major axis indicates the average magnitude along the corresponding timeline, while that of the minor axis keeps the same for all ellipses. We empirically choose the length of the minor axis so that all ellipses are visible and multiple ellipses can be placed along the river.

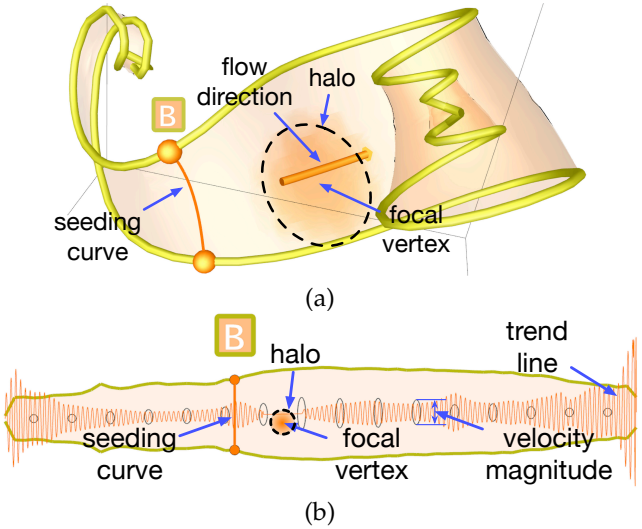


Fig. 7: Trend line using the five critical points data set. (a) a stream surface with curls at both ends. (b) the corresponding river with trend line and ellipse details.

Mini-map. As shown in Figure 6, we display a mini-map in the river view to provide a global picture of the rivers

when zoomed in. The actual position of the mini-map can be adjusted by users to avoid or reduce the overlap with the rivers drawn. The currently displayed range in the river view is shown by a bounding box, for which users can drag to pan around for quick navigation.

Branch labeling. A key challenge of examining stream surfaces is the difficulty in observing flow divergence in the convoluted 3D spatial view. We label the branches in both views to build the correspondence between the surface and river branches. Each river is hierarchically structured as a k-way tree with each node being a branch, as shown in Figure 10. We define the branch containing the seeding curve as the root of the tree. All labels' background color from the same surface/river uses the corresponding surface/river's color. The root branch is labeled using a single letter (e.g., B) in the same way as we label the entire river as both labels would not be displayed simultaneously. The label of a child branch appends a number to the label of its parent. For example, C1 denotes the first child branch of C, and D1-2 denotes the second child branch of D1.

How to read river view? We describe how to read the 2D river view and its encoded information to identify interesting features on the 3D stream surfaces. First of all, each river corresponds to a surface of the same color (Figure 6). The river always flows from left to right. The length of the river matches the extent of the surface (assuming the same integration method and step size are used) along the flow direction. One can loop through the timeline to show the correspondence between river segments and surface regions, along with the added labeling information. The river branching shows where and how the surface diverges (e.g., surface A in Figure 6). Look for where the river forms multiple branches (e.g., the branches near the seeding curve in Figure 11 (b)), indicating interesting and complex flow divergence around the surface. Branches on the same side of the seeding curve (e.g., branches D2 and D1-2 in Figure 10 (b)) imply that the corresponding flow joins (on the upstream side) or splits (on the downstream side). The trend line provides the average curvature and torsion information along the timeline (Figure 7). Look for high amplitude and frequency segments of the trend line for the corresponding surface regions with high curvature and torsion values (e.g., both ends of the surface in Figure 7). The ellipses along the trend line reveal velocity or vorticity magnitude variation along the corresponding timelines. Look for more stretched ellipses (e.g., Figure 7) for the corresponding surface regions with high velocity or vorticity magnitude.

5.2 Multilevel Highlighting

Besides standard brushing and linking that builds the connections between the surface and river views, we provide a set of highlighting functions to further enhance the mental connections and enable effective visual reasoning between the two views at the *river*-, *branch*-, and *vertex*-levels. Users specify these three levels of highlighting by mouse clicking while pressing the R, B, or V keys, respectively. Different highlighting levels can coexist except for highlighting a branch containing the seeding curve and highlighting the same river to which the branch belongs. When this happens, only the latest highlighting will be in effect.

River highlighting. When a river or its corresponding surface is highlighted, the river becomes the focal river (e.g., river D shown in Figure 6). The non-focal surfaces and rivers will be deemphasized by reducing the opacity in the surface view and color saturation in the river view. Furthermore, the non-focal rivers are automatically aligned with the focal one via similarity alignment. Specifically, for the similarity alignment energy term E_s, the focal river's constraints are given a higher weight. This allows the similarity between the focal and other rivers to be preserved at a higher priority. Note that this setting may consequently change timeline correspondence and vertical ordering.

Branch highlighting. When a river branch or its corresponding surface branch is highlighted, their boundaries in both views are highlighted. We add a boundary to the corresponding labels; the labels' boundary color follows that of the river/surface branch's boundary. In addition, the flow direction on the branch is displayed in the surface view. Similar to river highlighting, we deemphasize the non-highlighted surface and river branches by reducing opacity and color saturation, respectively. We allow users to simultaneously highlight multiple branches on the same river or different rivers for better referencing. For example, in Figure 10, multiple river branches A, A3, B, C1, C2, D1-2, and D2 are highlighted. To help examine the temporal evolution over a branch, we highlight in both views, the timeline under the cursor on this branch and the corresponding timelines on its sibling branches. By moving the mouse along the branch, users may observe how the timelines evolve on these sibling branches. Figure 11 shows such an example.

Vertex highlighting. While pressing the V key, users can click on a river to identify a focal vertex (when multiple rivers overlap, users simply mouse over to decide which one should be on top). They can also press the same key and click on a surface to specify the focal vertex in the maximum intensity projection manner (i.e., we take the vertex of the highest curvature value). The focal vertex is highlighted in both views, and the flow direction is marked on the focal vertex in the surface view. Figure 7 (b) shows a halo centered at the focal vertex in the river view. To best highlight the focal vertex, we automatically deemphasize the corresponding branch to which a focal vertex belongs regardless of the branch's highlighting status. In the surface view shown in Figure 7 (a), the transparency of a vertex on the surface is adjusted according to its geodesic distance to the focal vertex based on a sigmoid function. To approximate the geodesic distance efficiently, we run the computation on a mesh downsampled by two from the original surface mesh using Dijkstra's algorithm with the focal vertex as the single source. Similar to branch highlighting, multiple vertices on the same or different branches can be highlighted simultaneously.

6 Results and Discussion

6.1 Data Sets and Surface Sampling

Data sets and timing. We have explored several data sets listed in Table 1. The timing was collected on a PC with an Intel Core i9-9900K CPU running at 3.6GHz, 64GB main memory, and an NVIDIA GeForce RTX 2080 Ti graphics card

TABLE 1: The list of data sets experimented. The numbers reported for branches, timelines, and vertices are the average ones generated from each family or set of stream surfaces. The time is calculated by averaging results gathered from 150 runs of the energy minimization process.

data set	spatial dimension	# branches	# timelines	# vertices	horizontal alignment (sec)	vertical ordering (sec)	
						stress majorization	linear solver
Bénard flow	$128 \times 32 \times 64$	42	1,349	17,037	0.086	0.027	0.621
crayfish	$322 \times 162 \times 169$	94	1,794	43,490	0.130	0.038	2.409
five critical points	$51 \times 51 \times 51$	17	403	4,683	0.016	0.020	0.042
hurricane	$500 \times 500 \times 100$	13	387	16,998	0.041	0.037	0.227
square cylinder	$192 \times 64 \times 48$	38	1,207	27,143	0.071	0.022	0.508
tornado	$64 \times 64 \times 64$	5	553	13,780	0.019	0.029	0.076
vessel	$280 \times 260 \times 160$	42	1,836	120,911	0.145	0.022	1.749

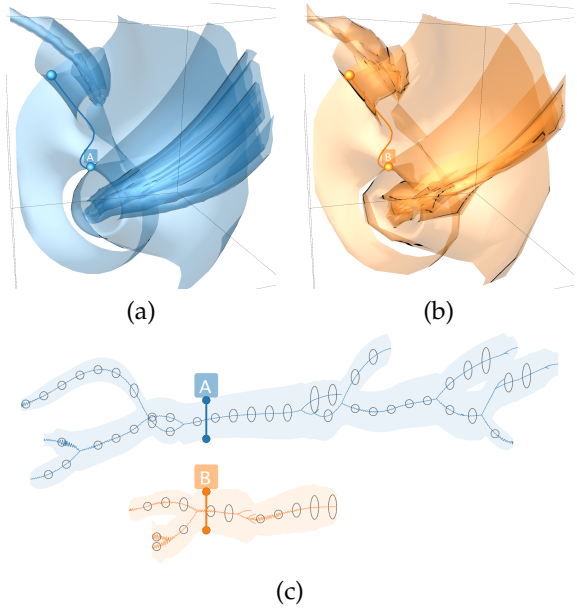


Fig. 8: SurfRiver of the five critical points data set. (a) and (b) stream surfaces traced from the same seeding curve using the fourth-order Runge-Kutta and first-order Euler methods, respectively. (c) the river view of these two surfaces with $\lambda = 91$ and $w_r = 0.96$.

with 11GB GDDR6 memory. A run of stress majorization takes about 100 iterations, and that of linear solver takes about 2,000 iterations. The time for vertical ordering using linear solver is larger than that using stress majorization. A set of downsampled data was fed to stress majorization to decrease the response time while the linear solver requires all of the data to achieve a smooth river layout.

Parameter setting. The desired river layout determines the choice of parameters. Although each data set's specific parameter settings may differ slightly, the general rule applies to most cases. For horizontal alignment, w_s is usually set to 0 since temporal alignment is our preferred alignment method. If similarity alignment is desired, setting w_s to 1 would achieve the most prominent similarity relations. A mix of temporal alignment and similarity alignment tends to average the effect parameter settings exert on the river layout. For vertical alignment, to attain a relatively straightening river layout (i.e., the river transits smoothly from timeline to timeline without experiencing sharp changes), we recommend a value between 0.95 and 0.99 for w_r. The more complicated (i.e., the complexity of timeline relations) the case is, the higher w_r value should be. If one wants to separate rivers (e.g., Figure 10), the value of λ should increase. A value in $[5, 100]$ for λ would satisfy, depending on the complexity of branches crossing with each other. For the rest of the paper, $\lambda = 0$ and $w_s = 0$ if not mentioned.

Surface sampling. Assume that we start with an existing seeding curve $p \rightsquigarrow q$, which is known to yield a suitable stream surface (e.g., borrowing the representative surface selection results from FlowNet [19]). Based on this seeding curve, we seek to create a group of seeding curves, which leads to a family of stream surfaces. There are multiple ways to do so. For example, we can fix one endpoint p, and sample on a small sphere centered at the other endpoint q. We can also rotate the seeding curve along its midpoint following a particular direction or within an angle range. Besides, we can lengthen or shorten the seeding curve either straightforwardly (for the seeding rake) or in a specific direction (e.g., the principal direction [27] or binormal direction [33]). Finally, all these could be generalized to anisotropic cases by considering the underlying flow properties such as critical point locations, flow entropy, or divergence.

6.2 Results

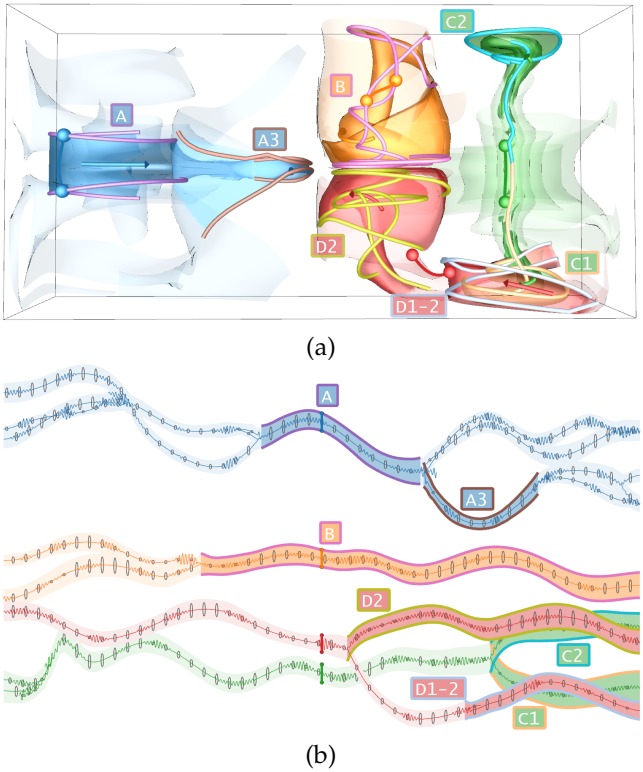
We discuss several cases with both representative surfaces and surface families to demonstrate the efficacy of SurfRiver. For the visual exploration of SurfRiver, please refer to the accompanying video.

Five critical points. In addition to mapping a family of stream surfaces that are more likely to be similar to each other (e.g., Figure 5), we also explore mapping a set of representative surfaces. The critical question is how to align multiple rivers for these surfaces as no standard reference exists now. In Figure 6, we show an example where the alignment of seven representative surfaces A to G is based on the seeding curves. Similarity alignment uses the MCPD of timelines. We can see that the spatial relationships among stream surfaces in 3D are preserved among rivers in 2D. For example, B, C, and D are neighbors, and B is in between C and D, which are consistent in both views. Furthermore, at the two ends of rivers B and D, their trend lines are rather bumpy (high amplitude and frequency). This indicates that the corresponding flow regions have high curvature and torsion values and are worth further exploration.

Figure 8 shows a side-by-side comparison of two stream surfaces generated from the same seeding curve but using different integration methods. The corresponding river view shows that the two rivers have very different extents. Even though both integration methods use the same step size, the first-order Euler method is less accurate in resembling the spirals, resulting in much fewer timelines. The river

(a) (b) (c)

Fig. 9: SurfRiver of the hurricane data set. (a) the surface view of two stream surfaces constructed from two vector fields with the same seeding curve. (b) the river view based on temporal alignment where $w_s = 0$. (c) the river view based on similarity alignment using JSD where $w_s = 1$. For (b) and (c), $\lambda = 89$ and $w_r = 0.98$.



(a) (b)

Fig. 10: SurfRiver of the Bénard flow data set showing four representative stream surfaces in (a) and the river view in (b) with $\lambda = 31$ and $w_r = 0.95$.

view also reveals the far more complex branching structure when tracing the surface using the fourth-order Runge-Kutta method, which is not noticeable by just comparing the surfaces.

Hurricane. Besides standard similarity alignment via MCPD, we can also align timelines according to their feature distributions using JSD. Such an example is given in Figure 9 using the hurricane data set. Instead of using a single time step of the data set, we now use the unsteady vector field's multiple time steps and place the same seeding curve to generate the stream surfaces from different time steps. For this example, the goal is to capture the trajectory of the hurricane's eye. The results with two stream surfaces are displayed here, where the orange/green surface corresponds to an early/later time step. For temporal alignment, as shown in Figure 9 (b), it is not apparent to link the hurricane's eye to the corresponding regions on both rivers because the eye's position shifts over time. For similarity alignment using JSD, as shown in Figure 9 (c), the black timeline on the green river is now automatically aligned with the orange timeline on the orange river because they have the most similar curvature and torsion patterns. Figure 9 (a) confirms that both timelines correspond to the evolving trajectory of the hurricane's eye at their respective time steps. Note that the displayed orange and black curves are *timelines* instead of *streamlines*. Therefore, these curves are not necessarily closed or form a helical configuration.

Bénard flow. Figure 10 shows the results of the Bénard flow data set using representative stream surfaces with several river branches highlighted. We find that it is easier to track the evolution of timelines using the river view. For example, in the surface view, we can only observe a swirling pattern on surface B. But in the river view, we can see that the seeding curve splits into two branches when tracing backward, suggesting that a diverging point is encountered when the seeding curve moves closer to the domain's boundary. As another example, surface D exhibits two swirls (i.e., branches D2 and D1-2). It is not clear whether the flow will leave one swirl and enter the other. The river view shows that both branches are formed by splitting the seeding curve along the flow. This implies that the flow will not move from one swirl to the other, as both branches appear on the same side of seeding curve in the river view.

Additionally, the layout of the rivers resembles the spatial relationships among the branches. For example, in the surface view, branch A3 is close to the other three surfaces. The river view shows a consistent layout, where A3 is pulled downward to stay close to the other rivers. C1 and C2 are two sibling branches deviated from C to the two sides of the domain's boundary in the surface view, or to the right end of the river view. Unlike C2, C1 continues to entangle with a newly involved branch D1-2, thus pulling these two branches closer.

Square cylinder. In Figure 11, we show the result of a single, complex stream surface from the square cylinder data set. We specifically place a seeding curve around the domain's center where the obstacle in the shape of a cylinder is located. As we can see, the seeding curve diverges quickly along both upstream and downstream directions, presenting intricate spatiotemporal relations. While the result is not apparent or even hidden in the surface view, SurfRiver can help us perform effective visual reasoning via the river view to glean insights. One insight we observe is that *temporally*

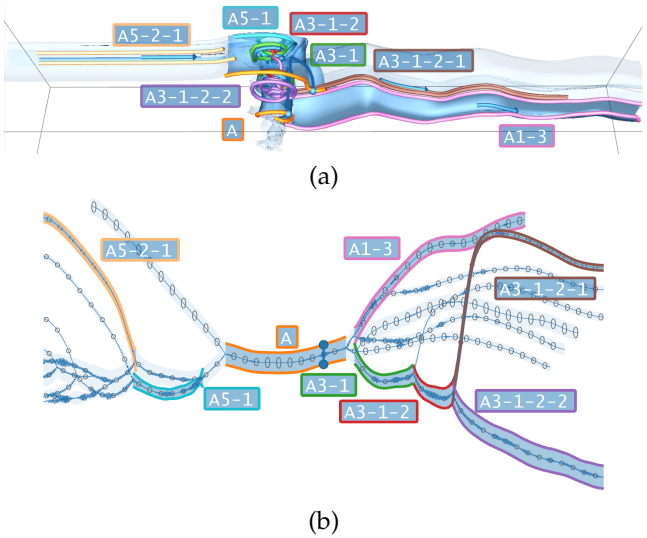


Fig. 11: SurfRiver of the square cylinder data set showing a stream surface in (a) with the seeding curve placed at the domain's center and the river view in (b) with $w_r = 0.97$.

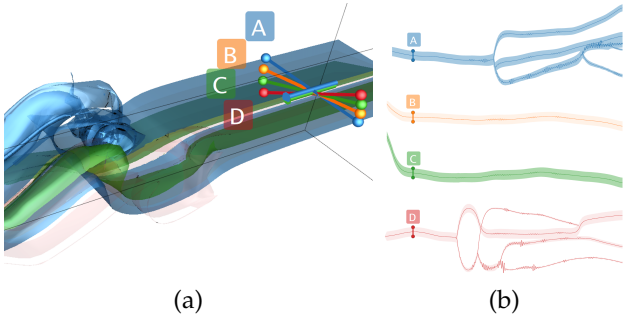


Fig. 12: SurfRiver of the square cylinder data set showing a group of slightly rotated seeding curves leads to very different stream surfaces in (a) and the river view in (b) with $\lambda = 43$ and $w_r = 0.95$.

close timelines could be spatially distant. For example, the timeline on branch A3-1-2-1, which is approaching the domain's boundary, shares an equal time step with the timeline on A3-1-2-2, which is still revolving at the domain's center. Another insight we observe is that *spatially close timelines could be temporally distant*. For example, the seeding curve on A is spatially close to the timeline on A3-1-2-2. However, A3-1-2-2 has experienced four diverging stages from the seeding curve (i.e., from A to A3 to A3-1 to A3-1-2 to A3-1-2-2), while remaining at the domain's center.

Furthermore, SurfRiver allows users to track the exact flow behavior in the swirl behind the cylinder. The surface view only reveals that the flow may leave the swirl in both the forward and backward directions, but it is impossible to track how many branches leave, when they leave, and whether there is still any branch staying in the swirl. In contrast, the river view delivers this information clearly. When examining the branches moving forward (on the right side of the river), we can see that branch A3-1-2-2 and its ancestors are distant from the other branches. By highlighting the branches, we find that A3-1-2-2 is the only branch that remains in the swirl. We can also easily track

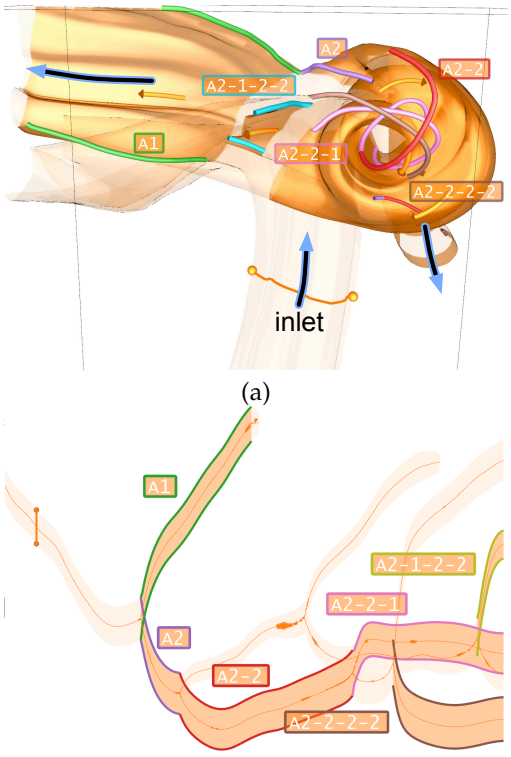
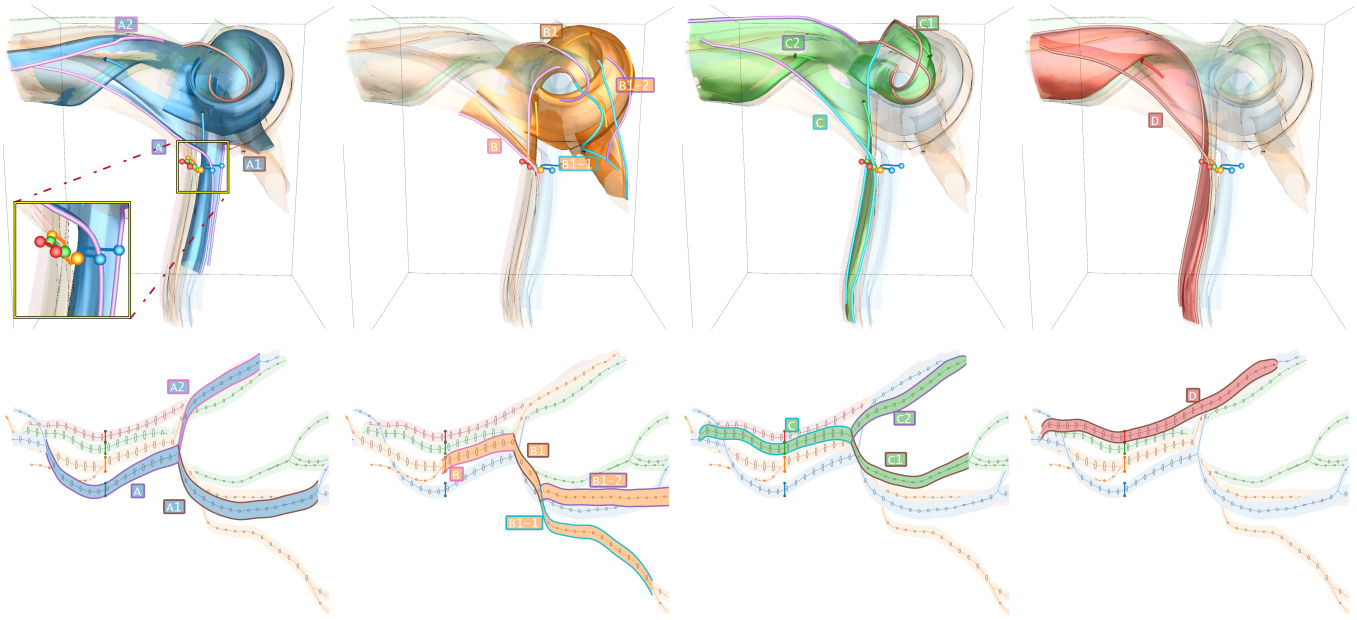


Fig. 13: SurfRiver of the vessel data set showing a stream surface seeded from the inlet in (a) and the river view in (b) with $w_r = 0.95$.

that branch A3-1 diverges twice, and two of its descendants are the last two branches that leave the swirl. All the other branches leave the swirl at a similar time, soon after the seeding curve is released. The branches moving backward show a similar pattern. The branches staying in the swirl appear at the lower part, while the ones leaving the swirl (for example, A5-2-1) move upward in the river view.

Figure 12 shows an example of using SurfRiver to study seeding sensitivity. Four seeding curves are generated from varying rotation angles along the same midpoint. We can see that these slightly different curves lead to very different surfaces. While surfaces B and C are simple, surfaces A and D are complex, forming interesting flow patterns after passing the obstacle. The corresponding four rivers confirm the observation and allow us to explore different branches of surfaces A and D.

Vessel. Figure 13 shows a single, complex stream surface seeded from the inlet of the vessel data set. As indicated by the flow directions shown in Figure 13 (a), the blood flow starts from the bottom and then bifurcates when encountering the aneurysm (at the top-right corner). After that, one flow branch moves to the left side, and another branch moves to the bottom-right corner of the aneurysm. Without SurfRiver, it could be challenging to examine the diverging flows that pass by and pass through the aneurysm, respectively. Even more challenging is the further divergences of these two flow branches. With SurfRiver shown in Figure 13 (b) and the linking to Figure 13 (a), it becomes apparent that surface A seeded at the inlet is separated into two branches A1 and A2. A1 completely misses the aneurysm, while



(a) (b) (c) (d)

Fig. 14: Looping through four stream surfaces (A to D) of the vessel data set for side-by-side examination of their similarities and differences. These surfaces are carefully seeded from the inlet close to the aneurysm. For the river views, $w_r = 0.97$.

A2 partially goes through the aneurysm. More specifically, within A2, A2-1-2-2 passes by the aneurysm, and A2-2, A2-2-1, and A2-2-2-2 pass through the aneurysm.

Figure 14 shows multiple stream surfaces produced from a series of seeding curves placed at the inlet and close to the aneurysm. As highlighted in the zoom-in view of Figure 14 (a), the seeding curves are produced by selecting a point near the aneurysm, cloning this point to three additional points along the normal direction, and extending each of these four points to a seeding curve along the binormal direction. As the seeding curves vary along the inlet, the stream surfaces exhibit the same general pattern of moving from the middle-bottom corner to the top-left corner. Closer examination finds that they are very different in terms of the respective flows concerning the aneurysm region. Surface D completely misses the aneurysm, while C covers only the inner part of the aneurysm (refer to branch C1). A covers a larger extent of the aneurysm than C. B covers an even larger extent of the aneurysm than A. Moreover, within the aneurysm, B1 is further diverged into B1-1 and B1-2. Unlike on the other surfaces, B1-1 and B1-2 flow into a different vessel branch at the middle-right corner. With SurfRiver, comparing and tracking these changes across multiple surfaces become easy and convenient for users.

6.3 Domain Expert Feedback

We collaborated with Dr. Jian-Xun Wang, an expert in computational fluid dynamics and a co-author of this work, to evaluate the effectiveness of SurfRiver. The evaluation consists of two stages, and each took about two hours. In the first stage, Dr. Wang was trained to learn the basics of SurfRiver and get familiar with the interface using the tornado and five critical points data sets. In the second stage, Dr. Wang was instructed to use SurfRiver to explore other data sets (the Bénard flow data set and the vessel data set provided by him) and complete a few predetermined tasks, such as changing seeding curves and identifying flow patterns. The following is a summary of his feedback.

Dr. Wang stated that "*Overall, SurfRiver is a useful tool to visualize and explore complex flow fields using stream surfaces. The most innovative aspect of this tool is the development of the 2D river view of the 3D surfaces.*" Compared to line-based solutions, surface-based approaches have their advantages. They can better represent complex flow structures for specific applications, e.g., hemodynamics, where the flow is less turbulent but very intricate due to geometric complexity. However, two main challenges prevent surface-based techniques from being widely adopted for flow visualization. First, it is unclear how to place seeding curves to generate and select flow surfaces effectively. Second, flow surfaces are more likely to create visual occlusion, making the analysis much tricky. He commented that "*The new 2D river view interface can help address these challenges. For example, in the 2D river view, users can efficiently study seeding sensitivity by interactively placing seeding curves and exploring the resulting surfaces. Moreover, using the 2D river view to explore 3D flow surfaces can largely eliminate visual occlusion, making surface-based flow visualization more accessible and usable.*"

Dr. Wang added that "*SurfRiver could be directly applied to cardiovascular flow visualization and analysis.*" The flow in human vasculature is often less turbulent (with moderate Reynolds number) yet has intricate flow patterns due to irregular vascular geometries. Thus, it can be well represented by stream surfaces. He further commented that "*The 2D river-based interface provides an effective way to extract meaningful flow structures by selecting out representative 3D flow surfaces in the 2D river map. Moreover, SurfRiver allows users to explore how the flow evolves and where the surfaces differ from each other in the 2D view. Lastly, SurfRiver visually encodes many typical flow features compactly and straightforwardly. The shape information (curvature and torsion) is clearly reflected as*

fluctuations of the trend line. The flow field characteristics (e.g., magnitudes of flow velocity or vorticity) are visually encoded as ellipses. This feature could be very useful in visualizing the multidimensional and multivariate hemodynamic information (e.g., flow velocity, pressure, vorticity, wall shear stress, etc.) simultaneously, providing a comprehensive analytical capability to better explore and understand the relation between hemodynamic factors and pathological developments."

Dr. Wang also pointed out some possible improvements for SurfRiver. First, although the seeding strategy is currently based on the user's manual selection and could provide sufficient flexibility, it also poses challenges in determining appropriate seeding curve placement. It might be helpful to provide users a candidate group of seeding curves that lead to representative flow surfaces, and users can select from and modify the seeding curve candidates to explore the flow more conveniently. Second, an extension of SurfRiver to path surfaces for unsteady flow will be exciting and useful in visualizing many transient flows. Third, the turbulence intensity information can be encoded into the trend line to reflect the flow's turbulent level.

6.4 Discussion

Overall, our experience, along with feedback from the domain expert, shows that SurfRiver does serve well its original purpose: enabling comparative visualization of a family of stream surfaces by flattening 3D surfaces into 2D rivers. Nevertheless, the current design and implementation have the following limitations. First, we can eliminate visual occlusion by adjusting λ for spatial proximity ordering. However, as Figure 4 shows, although displaying separated branches eliminates visual occlusion, the spatial relationships among 3D surface branches may no longer be preserved in the 2D river branches. Therefore, in practice, users may still prefer to maintain spatial relationships while sacrificing occlusion-free mapping. Second, unlike MCPD, we find it difficult to generate stable river layouts when using JSD for similarity alignment. This is because as the reference timelines l_i moves to its neighboring ones on a surface, the most similar timeline l_k on a different surface would likely change, leading to unstable alignment behaviors. This problem is more pronounced when we examine timelines around flow feature regions, such as the hurricane's eye, where the feature distributions for neighboring timelines could vary significantly. Third, there are some issues with the ellipse encoding (e.g., uniform sampling along the timelines may lead to potential loss of feature information), and more evaluation is needed. Fourth, although our current implementation automatically zooms in/out one view when the other view gets zoomed in/out manually, the minimap could also be automatically panned or re-centralized when the timelines of interest are about to get out of bound from the current view. Going further, we can integrate focus+context visualization into visual exploration as well.

7 Conclusions and Future Work

We have presented SurfRiver, an analytical solution for comparative visualization of stream surfaces by transforming them into the TextFlow-like visual representation. Flattening and aligning surfaces in 3D to rivers in 2D allows clear comparison of multiple stream surfaces. We demonstrate the effectiveness of SurfRiver in augmenting the user's ability to examining and reasoning seeding sensitivity and variability of a family of stream surfaces. The evaluation provided by a domain expert also confirms the efficacy of SurfRiver.

For future work, besides mapping the flow direction to the horizontal direction of SurfRiver, we will also explore mapping the orthogonal timeline direction to the horizontal direction, enabling the comparison of the variational flow information for a streamline along the seeding curve direction. The general idea of SurfRiver can be applied to flattening other surfaces, such as path surfaces and streak surfaces, for visual abstraction. But we should also note that the nature of unsteady flow field poses several challenges. First, unlike stream surfaces, path surfaces and streak surfaces may have self-intersections, leading to more complicated spatial patterns. Second, in unsteady flow fields, the temporal information of timelines carries physical meaning. Therefore, the spatiotemporal relationships between timelines must be considered, which increases the complexity as well. We will consider further constraints to appropriately resemble these patterns in 2D and provide additional hints for users to understand the temporal information on surfaces generated from unsteady flow fields.

Acknowledgements

This research was supported in part by the U.S. National Science Foundation through grants IIS-1455886, DUE-1833129, IIS-1955395, CMMI-1934300, Notre Dame Asia Research Collaboration Grant Program, the National Natural Science Foundation of China through grant 61902446, and the National Numerical Windtunnel Project. The authors would like to thank the anonymous reviewers for their insightful comments.

References

[1] P. Angelelli and H. Hauser. Straightening tubular flow for side-by-side visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2063–2070, 2011.

[2] T. Auzinger, G. Mistelbauer, I. Baclija, R. Schernthaner, A. Köchl, M. Wimmer, M. E. Gröller, and S. Bruckner. Vessel visualization using curved surface reformation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2858–2867, 2013.

[3] M. Bartoň, J. Kosinka, and V. M. Calo. Stretch-minimising stream surfaces. *Graphical Models*, 79:12–22, 2015.

[4] M. A. Borkin, K. Z. Gajos, A. Peters, D. Mitsouras, S. Melchionna, F. J. Rybicki, C. L. Feldman, and H. Pfister. Evaluation of artery visualizations for heart disease diagnosis. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2479–2488, 2011.

[5] S. Born, A. Wiebel, J. Friedrich, G. Scheuermann, and D. Bartz. Illustrative stream surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1329–1338, 2010.

[6] A. Brambilla, P. Angelelli, Ø. Andreassen, and H. Hauser. Comparative visualization of multiple time surfaces by planar surface reformation. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 88–95, 2016.

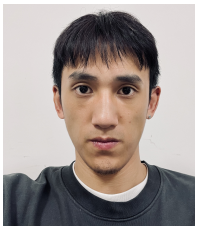
[7] A. Brambilla and H. Hauser. Expressive seeding of multiple stream surfaces for interactive flow exploration. *Computers & Graphics*, 47:123–134, 2015.

[8] W. Cai and P.-A. Heng. Principal stream surfaces. In *Proceedings of IEEE Visualization Conference*, pages 75–81, 1997.

[9] R. Carnecky, R. Fuchs, S. Mehl, Y. Jang, and R. Peikert. Smart transparency for illustrative visualization of complex flow surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 19(5):838–851, 2013.

[10] W. Cui, S. Liu, L. Tan, C. Shi, Y. Song, Z. J. Gao, X. Tong, and H. Qu. TextFlow: Towards better understanding of evolving topics in text. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2412–2421, 2011.

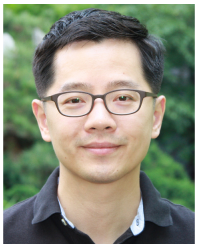
[11] M. Edmunds, R. S. Laramee, G. Chen, N. Max, E. Zhang, and C. Ware. Surface-based flow visualization. *Computers & Graphics*, 36(8):974–990, 2012.
[12] M. Edmunds, R. S. Laramee, R. Malki, I. Masters, N. Croft, G. Chen, and E. Zhang. Automatic stream surface seeding: A feature centered approach. *Computer Graphics Forum*, 31(3):1095–1104, 2012.
[13] J. M. Esturo, M. Schulze, C. Rössl, and H. Theisel. Poisson-based tools for flow visualization. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 241–248, 2013.
[14] E. R. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In *Proceedings of International Symposium on Graph Drawing*, pages 239–250, 2004.
[15] C. Garth, H. Krishnan, X. Tricoche, T. Bobach, and K. I. Joy. Generation of accurate integral surfaces in time-dependent vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1404–1411, 2008.
[16] C. Garth, X. Tricoche, T. Salzbrunn, T. Bobach, and G. Scheuermann. Surface techniques for vortex visualization. In *Proceedings of Eurographics - IEEE TCVG Symposium on Visualization*, pages 155–164, 2004.
[17] T. Günther, M. Schulze, J. Martinez Esturo, C. Rössl, and H. Theisel. Opacity optimization for surfaces. *Computer Graphics Forum*, 33(3):1725–1734, 2014.
[18] T. Günther, H. Theisel, and M. Gross. Decoupled opacity optimization for points, lines and surfaces. *Computer Graphics Forum*, 36(2):153–162, 2017.
[19] J. Han, J. Tao, and C. Wang. FlowNet: A deep learning framework for clustering and selection of streamlines and stream surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 26(4):1732–1744, 2020.
[20] J. P. M. Hultquist. Constructing stream surfaces in steady 3D vector fields. In *Proceedings of IEEE Visualization Conference*, pages 171–178, 1992.
[21] M. Hummel, C. Garth, B. Hamann, H. Hagen, and K. I. Joy. IRIS: Illustrative rendering for integral surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1319–1328, 2010.
[22] A. Kanitsar, D. Fleischmann, R. Wegenkittl, P. Felkel, and M. E. Gröller. CPR: Curved planar reformation. In *Proceedings of IEEE Visualization Conference*, pages 37–44, 2002.
[23] A. Kanitsar, R. Wegenkittl, D. Fleischmann, and M. E. Gröller. Advanced curved planar reformation: Flattening of vascular structures. In *Proceedings of IEEE Visualization Conference*, pages 43–50, 2003.
[24] O. D. Lampe, C. Correa, K.-L. Ma, and H. Hauser. Curve-centric volume reformation for comparative visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1235–1242, 2009.
[25] J. Ma, C. Wang, and C.-K. Shene. FlowGraph: A compound hierarchical graph for flow field exploration. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 233–240, 2013.
[26] J. Ma, C. Wang, C.-K. Shene, and J. Jiang. A graph-based interface for visual analytics of 3D streamlines and pathlines. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1127–1140, 2014.
[27] J. Martinez Esturo, M. Schulze, C. Rössl, and H. Theisel. Global selection of stream surfaces. *Computer Graphics Forum*, 32(2):113–122, 2013.
[28] T. McLoughlin, R. S. Laramee, and E. Zhang. Easy integral surfaces: A fast, quad-based stream and path surface algorithm. In *Proceedings of Computer Graphics International*, pages 73–82, 2009.
[29] T. Schafhitzel, E. Tejada, D. Weiskopf, and T. Ertl. Point-based stream surfaces and path surfaces. In *Proceedings of Graphics Interface*, pages 289–296, 2007.
[30] G. Scheuermann, T. Bobach, H. Hagen, K. Mahrous, B. Hamann, K. I. Joy, and W. Kollmann. A tetrahedra-based stream surface algorithm. In *Proceedings of IEEE Visualization Conference*, pages 151–158, 2001.
[31] M. Schulze, T. Germer, C. Rössl, and H. Theisel. Stream surface parameterization by flow-orthogonal front lines. *Computer Graphics Forum*, 31(5):1725–1734, 2012.
[32] M. Schulze, J. Martinez Esturo, T. Günther, C. Rössl, H.-P. Seidel, T. Weinkauf, and H. Theisel. Sets of globally optimal stream surfaces for flow visualization. *Computer Graphics Forum*, 33(3):1–10, 2014.
[33] J. Tao and C. Wang. Semi-automatic generation of stream surfaces via sketching. *IEEE Transactions on Visualization and Computer Graphics*, 24(9):2622–2635, 2018.
[34] J. Tao, C. Wang, N. V. Chawla, L. Shi, and S. H. Kim. Semantic flow graph: A framework for discovering object relationships in flow fields. *IEEE Transactions on Visualization and Computer Graphics*, 24(12):3200–3213, 2018.
[35] J. J. van Wijk. Implicit stream surfaces. In *Proceedings of IEEE Visualization Conference*, pages 245–252, 1993.
[36] V. Verma and A. Pang. Comparative flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):609–624, 2004.



Jun Zhang is a Ph.D. student at the University of Notre Dame. He received a B.E. degree in computer engineering from Sun Yat-sen University in 2019. His research interests are flow visualization and deep learning for scientific visualization.

Jun Tao is an associate professor of computer science at Sun Yat-sen University and National Supercomputer Center in Guangzhou. He received a Ph.D. degree in computer science from Michigan Technological University in 2015. Dr. Tao's major research interest is scientific visualization, especially on applying information theory, optimization techniques, and topological analysis to flow visualization and multivariate data exploration.

Jian-Xun Wang is an assistant professor of aerospace and mechanical engineering at the University of Notre Dame. He received a Ph.D. degree in aerospace engineering from Virginia Tech in 2017. Dr. Wang's research focuses on data-enabled computational modeling for fluid dynamics, including data-augmented or data-driven modeling framework and solutions for cardiovascular mechanics, turbulent flows, tsunami flows, and other general fluid dynamics problems.



Chaoli Wang is an associate professor of computer science and engineering at the University of Notre Dame. He received a Ph.D. degree in computer and information science from The Ohio State University in 2006. Technological University. Dr. Wang's main research interest is data visualization, in particular on the topics of time-varying multivariate data visualization, flow visualization, as well as information-theoretic algorithms, graph-based techniques, and deep learning solutions for big data analytics. He is an associate editor of IEEE Transactions on Visualization and Computer Graphics.