Peeling the Flow: A Sketch-Based Interface to Generate Stream Surfaces

Jun Tao* Chaoli Wang†

University of Notre Dame

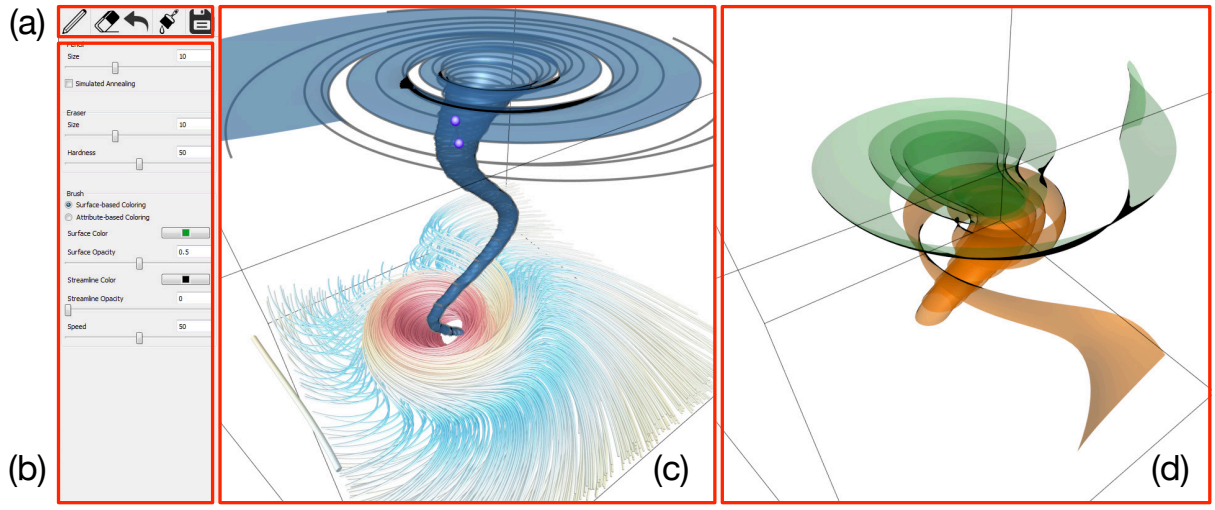


Figure 1: *(a) to (d) show the panel of tools, the panel of tool parameters, the streamline widget, and the stream surface widget, respectively. In (a), the five tool icons displayed from left to right are pencil, eraser, undo, brush and save, respectively. Users sketch to generate a new stream surface in (c). Once confirmed, the newly generated surface will be moved to (d) for visualization.*

Abstract

We present a user-centric approach for stream surface generation. Given a set of densely traced streamlines over the flow field, we design a sketch-based interface that allows users to draw simple strokes directly on top of the streamline visualization result. Based on the 2D stroke, we identify a 3D seeding curve and generate a stream surface that captures the flow pattern of streamlines at the outermost layer. Then, we remove the streamlines whose patterns are covered by the stream surface. Repeating this process, users can peel the flow by replacing the streamlines with customized surfaces layer by layer. Our sketch-based interface leverages an intuitive painting metaphor which most users are familiar with. We present results using multiple data sets to show the effectiveness of our approach, and discuss the limitations and future directions.

Keywords: Flow visualization, sketch-based interface, seeding curves, stream surfaces.

Concepts: •**Human-centered computing** → **Scientific visualization;**

*e-mail: jtao1@nd.edu

†e-mail: chaoli.wang@nd.edu

SA '16 Symposium on Visualization, December 05-08, 2016, Macao

ISBN: 978-1-4503-4547-7/16/12

DOI: http://dx.doi.org/10.1145/3002151.3002158

1 Introduction

In flow visualization, a stream surface is the integration of a 1D seeding rake or curve through a 3D steady flow field. The resulting surface is everywhere tangent to the local flow. As a natural generalization of streamlines, stream surfaces represent a continuum of streamlines. Besides indicating flow directions, stream surfaces can depict folding, shearing and twisting behaviors, enhance the visual perception of complex flow structures, and facilitate an intuitive understanding of flow geometry [Dallmann 1983; Garth et al. 2004].

Compared to streamline visualization, effective stream surface visualization is much more difficult to achieve. Unlike streamlines which can be uniquely determined by seeding locations, the seeding curves of stream surfaces have a much greater degree of freedom to vary. The differences in length, location and shape of seeding curves lead to different stream surfaces. In addition, the quality of stream surfaces is often difficult to predict, which requires more post analysis after the surface is generated. Existing works on surface seeding often allow users to specify two endpoints for generating the seeding rake. Seeds are placed on the resulting line segment for surface tracing. While straightforward, this approach limits the search space to the surfaces with at least one straight timeline (i.e., the seeding rake), which may not be ideal for capturing flow patterns. Moreover, it is difficult for users to adjust the endpoints to improve the quality of the generated stream surface, since the adjustment result is hardly foreseeable. Therefore, fine tuning the seeding rake normally entails painstaking trial-and-error efforts.

Inspired by the concept of "structure from motion", which reconstructs 3D structures from a set of 2D images [Lowe 2004], we conjecture that good seeding curves can be inferred from a set of streamlines. We can randomly place seeds to produce a set of streamlines that densely cover the entire domain. Unlike discrete vectors at isolated voxel locations, integral streamlines give con-

tinuous impression of the underlying flow over the domain and thus provide more immediate hints to search for good seeding curves. As a matter of fact, human perception can naturally pick up appropriate surfaces from a given set of streamlines. Our goal is try to mimic this process. In this regard, our work is in line with other semi-automatic techniques that leverage human knowledge for machine-challenging tasks. For example, users may draw strokes on an image to indicate the foreground and background for segmentation [Boykov and Jolly 2001].

Following the above notion, we present a sketch-based interface that allows users to generate desired stream surfaces on top of a set of densely traced streamlines. Users can simply draw a 2D stroke directly on the streamline visualization result to indicate the favored seeding curve. We derive a binormal field from the flow field and identity a 3D seeding curve following the binormal direction. The 2D projection of this seeding curve is similar to the user-drawn stroke and it leads to a surface that captures the flow pattern of the outermost layer of streamlines (i.e., the ones that are closest to users along the viewing direction). Once the stream surface is generated, the streamlines whose distances to the surface are smaller than a user-defined threshold will be removed. Repeating this process, users can "peel" the streamlines layer by layer, and each layer becomes one stream surface. During this process, users can simply remove the less interesting streamlines as well. They can further edit a surface by dragging the endpoints of the seeding curve to extend the surface for a more complete coverage, or to shrink the surface for a more concise representation. They can also adjust the rendering effect to achieve desirable surface visualization results. All these functions are provided by an intuitive painting interface where users draw seeding curves using the pencil tool, drag a seeding curve using the hand tool, remove streamlines using the eraser tool, and adjust surface rendering using the brush tool.

2 Related Work

Surface-Based Flow Visualization. Existing stream surface generation techniques include extracting stream surfaces as isosurfaces in specifically designed scalar fields [van Wijk 1993; Cai and Heng 1997], generating seeding curves as isolines of scalar fields on domain boundaries [Edmunds et al. 2012b], generating seeding curves based on streamline clusters [Edmunds et al. 2012a], and generating seeding curves based on a seeding structure (e.g., a seeding plane) [Sadlo et al. 2004; Brambilla and Hauser 2015].

Hultquist [Hultquist 1992] presented the first work for surface construction that advances a seeding front to generate stream surfaces. Several works further refine the seeding fronts during advancement, such as arc-length-based [Garth et al. 2004], quad-based [McLoughlin et al. 2009], and point-based [Schafhitzel et al. 2007] algorithms. Other works improve the interpolation using tetrahedral grids [Scheuermann et al. 2001] or Hermite interpolation [Schneider et al. 2009]. Garth et al. [Garth et al. 2008] advocated a two-step surface generation. It first generates a skeleton of the integral surface, followed by a well-conditioned triangulation.

Surface rendering aims at reducing visual occlusion and clutter, and enhancing depth and spatial perception of flow features and structures. Existing techniques leverage contour lines and half-toning [Born et al. 2010], transparency and texturing [Hummel et al. 2010], and illustration buffer [Carnecky et al. 2013] to improve and enhance the perception of surfaces.

Closely related to our work are the works of global selection of single and multiple stream surfaces [Martinez Esturo et al. 2013; Schulze et al. 2014]. Martinez Esturo et al. [Martinez Esturo et al. 2013] proposed to favor surfaces where the flow is aligned with principal curvature directions. They leveraged simulated annealing to select a globally optimal stream surface based on a set of stream surface quality measures. Schulze et al. [Schulze et al. 2014] extended the work to select a set of globally optimal stream surfaces in an iterative manner. All selected surfaces are mutually distant and optimize global stream surface quality measures.

The two solutions above are fully automatic. Therefore, they could not be customized according to user intentions or needs. Due to the highly flexible nature of seeding curves and the huge candidate pool of stream surfaces, seeking a unique, optimal set of surfaces demands the generation of an excessive number of stream surfaces for goodness test. Instead, we advocate a user-centric approach and allow users to sketch 2D strokes directly on top of the streamline visualization result to specify surfaces of interest. Our reasoning is that users can play an instrumental role in this challenging task by conveying their intuition (i.e., where to place the surfaces) and priority (i.e., the order of surfaces created) to quickly narrow down the search space. In this way, we are able to produce comparable stream surface results cost-effectively, even though we do not claim that the surfaces are optimal.

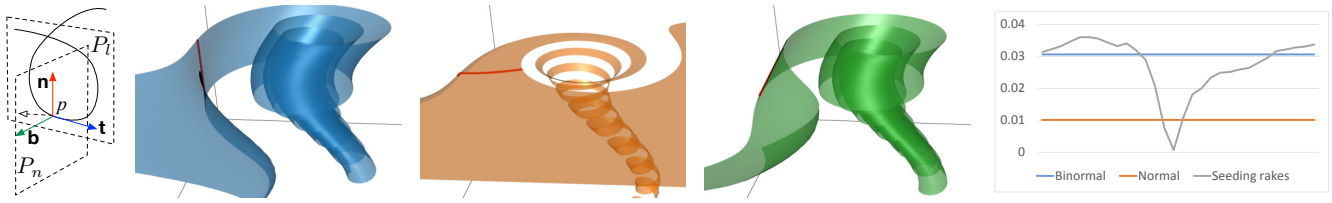
Sketch-Based Interface for Scientific Visualization. Sketch- and touch-based interface and interaction have been introduced to assist volume visualization, including transfer function design [Tzeng et al. 2003], WYSIWYG volume visualization [Guo et al. 2011], and Visualization-by-Sketching for time-varying multivariate data visualization [Schroeder and Keefe 2016]. For flow visualization, sketch-based interfaces have been applied to 2D illustrative visualization [Schroeder et al. 2010], 3D flow field classification [Wei et al. 2010], and exploration of scientific data sets [Klein et al. 2012]. Unlike the sketch-based interfaces presented in [Schroeder et al. 2010; Wei et al. 2010] for streamline visualization, we target the more challenging problem of stream surface generation. For convenience, we allow users to directly draw 2D stokes on top of the streamline visualization result to specify seeding curves, rather than painting on 2D slices of the volume in a separate view [Tzeng et al. 2003]. Compared to the seeding rake generation [Klein et al. 2012], we do not require a cutting plane for user sketching. Instead of directly using the user's input as the seeding curve, we identify a seeding curve that leads to an improved surface with good quality while matching the user's intention.

3 Stream Surface Generation

Our approach identifies a seeding curve based on a user-drawn stroke, so that the stream surface generated from the seeding curve covers the outermost layer of streamlines intersecting with the stroke. To capture the flow pattern, the stream surface should align with the flow. That is, a local patch of the surface lies in the same plane defined by the *tangent* and *binormal* vectors at a point on the streamline. The relationship among the tangent $\mathbf{t}$, normal $\mathbf{n}$, and binormal $\mathbf{b}$ vectors is shown in Figure 2 (a). For a point p on the streamline, $\mathbf{t}$ and $\mathbf{n}$ lie on the plane of P_l which contains the streamline segment centered at p, while $\mathbf{b}$ is perpendicular to P_l. Since streamlines on a stream surface follow the flow direction, the timelines are preferred to follow the binormal direction. We approximate a user-drawn stroke with a seeding curve following the binormal direction and trace the surface using the quad-based approach [McLoughlin et al. 2009].

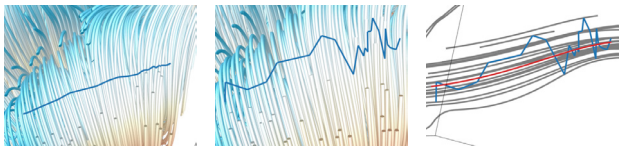
3.1 Aligning Seeding Curve along Binormal Direction

The reason for us to use the binormal direction to align a seeding curve is based on the following observation. It is ideal to generate seeding curves that are as *perpendicular* to the flow as possible, since this kind of seeding curve would maximize the effective length to generate stream surfaces. In Figure 2 (a), the plane P_n is



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

Figure 2: *(a) the tangent* **t***, normal* **n***, and binormal* **b** *vectors at a point p on a streamline. (b) to (d) show stream surfaces seeded along the binormal direction, the normal direction, and the best straight seeding rake, respectively. (e) quality measure using the average squared normal curvature.*



(a) (b) (c)

Figure 3: *Seeding curve approximation based on a user-drawn stroke. (a) shows the stroke under the original viewpoint. (b) shows the stroke under another viewpoint, so that its 3D shape can be perceived. (c) shows the stroke in blue, the candidate seeding curves in gray, and the actual seeding curve in red.*

defined by the normal **n** and binormal **b** vectors. P_n is perpendicular to the flow direction (i.e., the tangent vector **t**). Obviously, there are still an infinite number of directions to take for the seeding curve on P_n. But we only consider the normal and binormal directions of the streamline, since they form an orthogonal basis of P_n. The normal direction **n** lies in the same plane of P_l as the streamline segment. Therefore, seeding along **n** is likely to create a planar surface. On the contrary, seeding along the binormal direction **b** is more desirable, since it leads to a surface with a more interesting 3D shape.

Using the tornado data set, we shown in Figure 2 (b) and (c) two stream surfaces traced from two seeding curves following binormal and normal directions, respectively. We can see that the surface seeded along the binormal direction better captures the 3D shape of the vortex core. In contrast, the surface seeded along the normal direction contains a large planar portion around the tail of the spiral. Even for the spiral, this surface is mostly perpendicular to the vortex core and it fails to convey the impression of a layer of the flow.

For a quantitative study, we leverage the normal curvature measure proposed by Martinez Esturo et al. [Martinez Esturo et al. 2013] to evaluate the quality of a surface. They measured the normal curvature of the surface at a point p, using the normal curvature

$$\frac{\mathbf{n_s}^T \mathbf{J}\mathbf{v}}{|\mathbf{v}|}, \tag{1}$$

where $\mathbf{n_s}$ is the surface normal at p, and $\mathbf{J}$ and $\mathbf{v}$ are the Jacobian matrix and flow velocity at p, respectively. Note that $\mathbf{v}$ is in the same direction as $\mathbf{t}$ with additional magnitude information. For fair comparison, we compute the average squared normal curvature of each surface by averaging the squared normal curvature over the entire surface. The stream surfaces that *minimize* this term are usually flat ones with vanishing normal curvature and produce less interesting results, and the surfaces that *maximize* this term often demonstrate a better quality. In addition to the two seeding curves along the binormal and normal directions, we evenly sample 30 straight lines pointing in different directions as seeding rakes for comparison. All these seeding rakes are centered at the midpoint p_c of the seeding curve following the binormal direction and reside in the plane P_n determined by the binormal and normal vectors at p_c. We make their lengths equal to the maximum distance between any two seeds on the binormal seeding curve, so that they cover a similar range as the seeding curve following the binormal direction does.

We plot the average squared normal curvature measured on the sampled seeding rakes as a gray curve in Figure 2 (e), starting from the rake following the binormal direction. The average squared normal curvature of surface generated from seeding along the binormal/normal direction is plotted as the blue/orange line. We can see that seeding along the binormal direction, the generated surface has close to best quality compared to the sampled seeding rakes. This is confirmed by comparing Figure 2 (b) and (d), as the two surfaces only differ by a small degree at the tails.

In practice, instead of computing the binormal directions from streamlines, we precompute a *binormal vector field* from the given vector field, so that the binormal direction at any point in the domain can be retrieved directly from the binormal vector field. The binormal direction **b** at a point p is given by

$$\mathbf{b} = \mathbf{n} \times \mathbf{t} = \mathbf{J}\mathbf{t} \times \mathbf{t}, \tag{2}$$

where **t**, **n** and **J** are the tangent direction, normal direction, and Jacobian matrix at p. The seeding curve is integrated in the binormal vector field using the fourth-order Runge-Kutta method. This is similar to tracing streamlines in the original vector field.

3.2 Seeding Curve Approximation

The seeding curve approximation is performed as follows. First, we map the points on the 2D user-drawn stroke back to the 3D space. For each point p, we identify the streamline that is first hit at p, and use the hit point as the mapped 3D point. Figure 3 (a) shows a user-drawn stroke in the original viewpoint, under which the stroke is sketched. Each point on this stroke is mapped to a point on a streamline that is closest to the screen under the original viewpoint. Observing from another viewing direction as shown in Figure 3 (b), we can see that although the original stroke seems relatively smooth, the 3D stroke after mapping is actually zig-zag due to the depth discontinuity among the streamlines.

Second, instead of smoothing this 3D stroke, we search for a curve which follows the binormal direction and is closest to the 3D stroke. For each point p on the 3D stroke, we generate $k+1$ seeds. Among these $k+1$ seeds, one seed is p itself, and the other k seeds are evenly spaced on a small sphere centered at p. Then, we trace $k+1$ curves from those seeds in the binormal vector field, and compute the mean of closest point distances from the 3D stroke to each curve. The closest curve is defined as the one with the smallest distance among all the generated curves. The mean of closest point

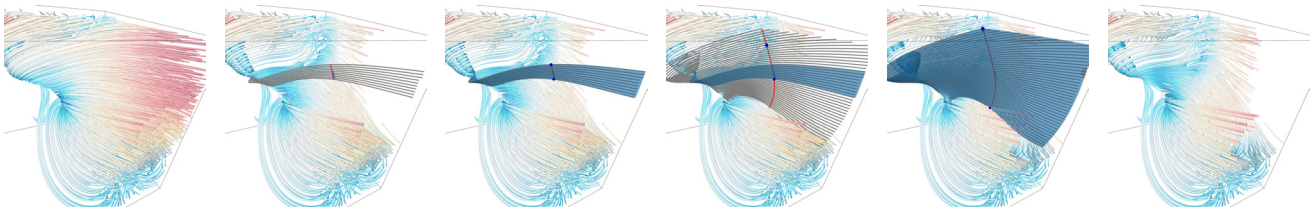


Figure 4: *Generating a single stream surface. (a) shows the original streamlines. (b) shows a user-drawn stroke in blue, the corresponding seeding curve in red, and guiding streamlines in gray. (c) shows the stream surface generated from the seeding curve. (d) shows the extended seeding curve and the corresponding guiding streamlines. (e) shows the stream surface generated from the extended seeding curve. (f) shows the remaining streamlines after removing the outermost layer of streamlines that is captured by the newly generated stream surface.*

distances from one point set $\mathbf{P} = \{p_1, p_2, \ldots, p_n\}$ to another point set $\mathbf{Q} = \{q_1, q_2, \ldots, q_m\}$ is formulated as follows

$$d(\mathbf{P}, \mathbf{Q}) = \frac{\sum_{p_i \in \mathbf{P}} \min_{q_j \in \mathbf{Q}} d(p_i, q_j)}{n}, \tag{3}$$

where $d(p_i, q_j)$ is the Euclidean distance between p_i and q_j. In Figure 3 (c), one gray curve is generated for each point on the blue 3D stroke, and the closest curve is the one highlighted in red. We can see that the depth randomness of streamlines is canceled out and the closest curve is approximately the centerline of the 3D stroke.

Third, since we trace the curves in the binormal vector field as long as possible, these curves are usually longer than the stroke itself. Therefore, we cut one segment on the curve that best fits the length of the stroke. Specifically, starting from the point that is closest to the 3D stroke, we scan the curve in both the forward and backward directions to identify two points that are closest to the two endpoints of the stroke. The segment between the two points identified is the seeding curve. In Figure 3 (c), the red curve segment is the seeding curve we generate for this example.

4 Interface and Interaction Design

As shown in Figure 1, our interface consists of four components: the tool panel, parameter panel, streamline widget, and stream surface widget. We provide six tools (pencil, eraser, hand, undo, brush and save) for users to complete different tasks in the two widgets. The streamline widget displays all streamlines and works with the pencil, eraser, hand and undo tools. It also displays the newly generated stream surface before users confirm the surface and move it to the stream surface widget. The stream surface widget displays all the confirmed stream surfaces. Users can use the brush tool to change their rendering effects. The save tool is used to output the current screenshot as an image file. In this section, we describe the widgets, and discuss the associated tools and interactions.

4.1 Streamline Widget

In the streamline widget, users sketch with the pencil tool to generate seeding curves and then create stream surfaces. They use the eraser tool to remove unimportant or surrounding streamlines, so that the stream surfaces covering the important or inner flow pattern can be generated subsequently. To modify the width of the stream surface, users can leverage the hand tool to drag the endpoints of its seeding curve along the binormal direction. We do not present an icon for the hand tool on the interface as this tool will be automatically enabled for fine tuning the seeding curve and surface right after the initial surface is generated.

Typical Workflow. We demonstrate the use of the streamline widget with an example shown in Figure 4. We start with a pool of

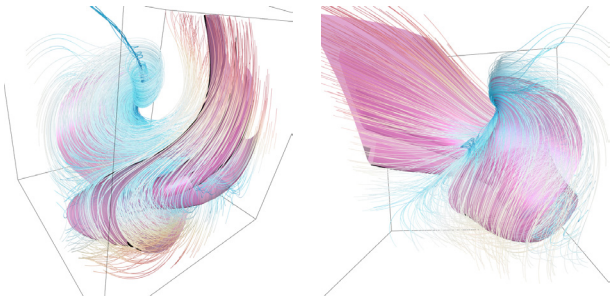


Figure 5: *A stream surface and its corresponding streamlines, determined by the mean of closest point distances. (a) and (b) show the surface and streamlines under two different viewpoints.*

streamlines densely traced over the field, as shown in (a). Users can use the eraser tool to remove less interesting streamlines and use the pencil tool to draw a stroke on top of the streamline visualization result to indicate the flow pattern they want to capture. We trace a set of *guiding streamlines* from evenly spaced seeds on this seeding curve to indicate the shape of corresponding stream surface. For a clearer observation, the streamlines close to the guiding streamlines will be temporarily removed, as shown in (b). Users can generate a stream surface from the seeding curve, if they feel that the guiding streamlines represent the desired flow pattern; otherwise, they can ignore this seeding curve by simply drawing another stroke.

After the stream surface is generated, the surface along with the two endpoints of its seeding curve will appear in the streamline widget, as shown in (c). Users can use the hand tool to drag the endpoints to extend or shrink the surface. When users finish dragging the endpoints, they can regenerate the surface, as shown in (e). If users are satisfied with the quality of the surface, they can confirm the surface and add it into the final visualization result. Once the stream surface is confirmed, it will be moved to the stream surface widget, and the streamlines whose pattern is captured by this surface will be removed from the streamline widget. Compared to the streamlines before surface generation as shown in (a), we can see that the outermost layer of streamlines is peeled, as shown in (f). Repeating this process, users can peel the flow field layer by layer, and generate the desired stream surfaces for visualization.

Tool Tuning. Several parameters can be adjusted for the pencil and eraser tools. For the pencil tool, the only parameter is its size. Similar to that in a painting interface which decides the thickness of stroke, the size of the pencil in our interface determines the thickness of the layer being peeled. The streamlines close to the guiding streamlines or the stream surface are peeled based on the mean of

closest point distances. The size parameter serves as the distance threshold to control which streamlines to remove. If the value is small, then only the streamlines that are very close to the guiding streamlines or the stream surface will be removed, which means that only a thin layer will be peeled. In Figure 5, we show a stream surface and the corresponding streamlines removed under two different viewpoints. We can observe that the streamlines approximately form a layer centered at the surface.

For the eraser tool, two parameters are available: size and hardness. The size parameter determines the radius of a circle centered at the mouse position on the screen. Under the current viewpoint, we remove the front streamlines that intersect with this circle in the screen space. The hardness parameter decides how many streamlines to remove at the mouse position. If the value is one, only the frontmost streamline will be removed at each sample position. We also provide the undo tool, so that users can click on the icon to recover the streamlines removed by the eraser tool.

4.2 Stream Surface Widget

In the stream surface widget, we render the stream surfaces and allow users to adjust rendering effects with the brush tool. The rendering effects include color, transparency, silhouette, and displaying streamlines on the surface. The silhouette visually emphasizes the transition between front- and back-facing surface layers, and facilitates better perception of surface shape. So it is applied to all surfaces by default. Other rendering effects can be adjusted by users. We provide two coloring schemes: surface-based coloring and attribute-based coloring.

Surface-Based Coloring. The surface-based coloring scheme uses different colors for different stream surfaces, but all points on the same surface share the same color. Using this scheme, different surfaces can be better distinguished. As suggested by Hummel and Garth [Hummel et al. 2010], the transparency is designed as a function of normal variation in the image space to emphasize surface details, such as ridges and valleys. Formally, the transparency value is given by $\alpha_v = v^{\gamma/2}$, where v is the normal variation and γ is a parameter in $[0,1]$.

The brush tool in this coloring scheme has three parameters: color, γ, and transparency factor t. The color can be selected from a color palette, while γ and t can be selected with sliders or input from text boxes. We use t to adjust the transparency of the entire surface to deemphasize the less important surfaces and emphasize the more important ones. The final transparency is defined as

$$\alpha = t((1-\alpha_v)\alpha_{\min} + \alpha_v\alpha_{\max}), \tag{4}$$

where $\alpha_{\min}$ and $\alpha_{\max}$ are used to constrain the transparency within a certain range. Using the brush tool, users can simply click on a surface to apply the current parameter values of the brush.

Attribute-Based Coloring. The attribute-based coloring scheme assigns the same color and transparency to surface points with the same attribute value. The scalar attribute, such as the curvature or torsion field derived from the flow field, is specified by users. This coloring scheme helps to distinguish flow patterns with a certain property as indicated by the selected attribute. Initially, the color at each surface point is given by a color map, and following Equation (4), its transparency is given by

$$\alpha = (1-a^{\gamma/2})\alpha_{\min} + a^{\gamma/2}\alpha_{\max}, \tag{5}$$

where a is the normalized attribute value. Under this scheme, two parameters can be adjusted for the brush tool: color and transparency. Users can use the brush tool to paint colors directly on top of the stream surface rendering. This will blend the brush color with the original color. When the brush tool is used to sketch on the surfaces, we accumulate the attribute values in the brush strokes. We partition the attribute values into a number of discrete ranges. Let n_i be the count of the i-th attribute range that is brushed by users. We update $c(i)$, the color of the i-th attribute value range, to

$$c(i) = (1-w_i)c_o(i) + w_i c_b, \tag{6}$$

where $c_o(i)$ is the original color of the i-th attribute value range, c_b is the brush color, $w_i = \min(n_i/N, 1)$, and N is a constant. If n_i is zero, which means that the i-th attribute value range is never brushed, that attribute value range uses its original color. If n_i is larger than N, the color of the i-th attribute value range becomes the brush color. The transparency is updated similarly.

data set	dimension	avg. # p/l	SR	timing (in seconds) curve fitting	surface integral	line removal
electron	$64\times64\times64$	58.0	1	0.43	0.19	0.16
five CPs	$51\times51\times51$	116.2	5	0.28	0.29	0.55
sq. cylinder	$192\times64\times48$	195.9	1	0.69	2.37	0.41
tornado	$64\times64\times64$	292.4	1	0.32	0.18	0.78
two swirls	$64\times64\times64$	918.2	10	0.48	3.10	0.69

Table 1: *Timing and parameters for each data set. We use 500 streamlines for the two swirls data set and 3000 streamlines for the other data sets. The timing reported is the average cost to generate one stream surface in the results. "avg. # p/l" is the average number of points per streamline and "SR" is sampling rate.*

5 Results

5.1 Performance and Parameters

We evaluated the performance of our approach using the data sets listed in Table 1. All results were collected on a PC with an Intel Core i7-4790 quad-core 3.60GHz CPU, 32GB main memory, and an nVidia Geforce 970 graphics card with 4GB graphics memory.

For most data sets, we randomly traced 3000 streamlines to fill the entire domain. For the two swirls data set, since the streamlines are relatively long with repeated patterns, we used less streamlines for better speed performance. The sampling rate r indicates that a point in every r points on a streamline is used to compute the mean of closest point distances between that streamline and a stream surface or a guiding streamline. This distance was used to determine whether or not a streamline close to a stream surface or the guiding streamlines should be removed. The streamline removal was performed in the GPU using CUDA, and its running time mainly depends on the numbers of points on a streamline and the stream surface. With the continuity of streamlines, we felt that the distances computed using the sampled points still produce reasonable results, especially for the long streamlines with repeated patterns.

For each data set, we generated as many surfaces as needed, until most streamlines were removed and the flow field was mostly captured by the surfaces. The curve fitting time includes the time to trace lines in the binormal vector field, fit the user's sketch to those lines, trace the guiding streamlines, and temporarily remove streamlines for occlusion reduction. The streamline tracing and removal were performed in the GPU, and the other steps were performed in the CPU. The guiding streamlines contain a much less number of points than the final surface, especially when the flow is complex. Therefore, removing streamlines according to the guiding streamlines is less costly.

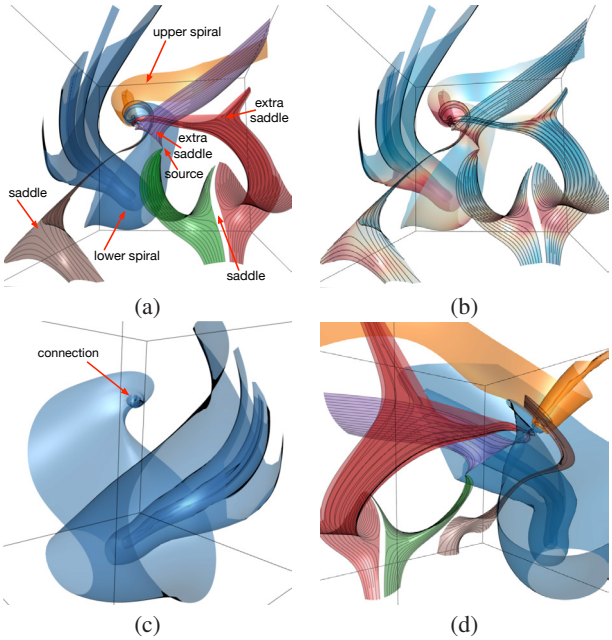


Figure 6: *(a) and (b) show six stream surfaces of the five critical points data set with surface-based coloring and attribute-based coloring, respectively. (c) shows the single blue stream surface. (d) shows the surfaces as seen from the opposite viewpoint of (a).*

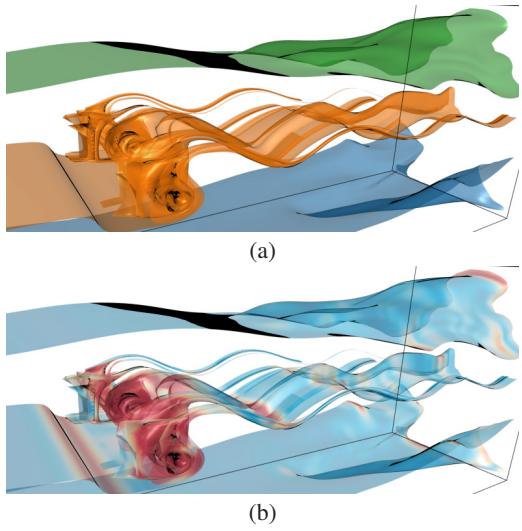


Figure 7: *(a) and (b) show three stream surfaces of the square cylinder data set with surface-based coloring and attribute-based coloring, respectively.*

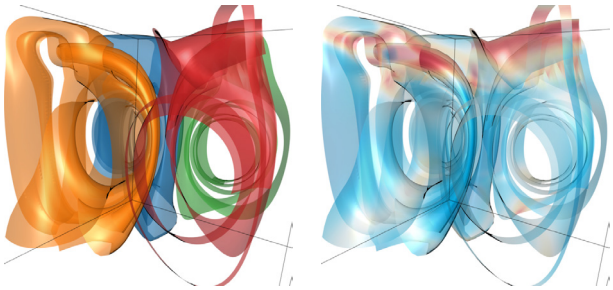


Figure 8: *(a) and (b) show four stream surfaces of the two swirls data set with surface-based and attribute-based coloring, respectively.*

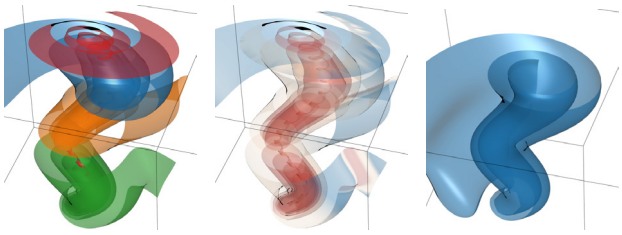
5.2 Visualization Results

Five Critical Points. The six stream surfaces we generate from the five critical points data set are shown in Figure 6. This data set is synthesized with five randomly generated critical points: two spirals, two saddles, and one source. The flow patterns related to these critical points and their connections are essential for understanding. We start from the larger spiral pattern at the corner and generate the blue stream surface, since it is the most obvious feature as seen from the outermost layer of streamlines. The blue surface covers a large portion of the field, revealing not only the flow pattern of the larger spiral, but also the connection between the two spirals, as highlighted in (c). Then, we sketch on the other spiral to generate the orange surface, and the two saddles to generate the green, red and brown surfaces. Finally, we generate the purple surface to fill the gap between the upper spiral and the source, where the green surface starts. The stream surfaces with surface-based coloring are shown in (a). We find that the flow directions on the spirals are easy to follow, but those on the other critical points may not be easily perceived at the first glance. We therefore add streamlines to enhance the perception of these critical points. The saddle between the green and red surfaces can be observed clearly. The saddle at the bottom left corner and the source near the center of the volume are captured by the streamlines, but their patterns are not outstanding. In (b), we switch to attribute-based coloring. The critical points can be noticed easily when the curvature attribute is used, since all critical points correspond to the high-curvature regions shown in red. We also find two extra saddles on the purple surface between the upper spiral and the source, and on the upper region of the red surface, by simply looking for all red regions. The connections among critical points can be observed clearly as well. A stream surface with multiple red regions is likely to connect different critical points. We can see that the blue surface connects the two spirals, the green surface connects the source and the bottom right saddle, the red surface connects the upper spiral and two saddles on the right, the brown surface connects the spiral and the bottom left saddle, and the purple surface connects the upper spiral, the source and one saddle between them. In (d), we render the stream surfaces from the opposite viewpoint to observe the two spirals and the connection between the source and the upper spiral.

Other Data Sets. The visualization results for the other data sets are shown in Figures 7 to 9. We generate three surfaces for the square cylinder data set to capture the upper, middle and lower layers of the flow (Figure 7), and four surfaces for the two swirls data set to capture the flow patterns in the front and back halves of each swirl (Figure 8). In Figure 9 (a), we show four surfaces for the tornado data set. To better reveal the inner pattern, we use the brush tool to adjust the opacity so that the low entropy regions become more transparent, as shown in Figure 9 (b). The flow pattern can be captured by a single surface as well, as shown in Figure 9 (c). Although the single surface in (c) provides a more compact visualization result, the four surfaces in (a) with their tails pointing into different directions reveal additional information of the flow field.

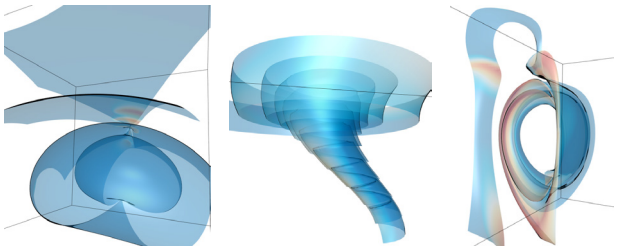
6 Discussion and Future Work

Seeking Optimal Seeding Curves. We approximate the user's sketch to generate the seeding curve by following the binormal di-



(a) (b) (c)

Figure 9: *(a) shows four stream surfaces of the tornado data set with surface-based coloring. (b) shows the surfaces with attribute-based coloring and edited color mapping. (c) shows one single stream surface that covers the entire domain.*



(a) (b) (c)

Figure 10: *The alignment errors of three stream surfaces from the (a) electron, (b) tornado, and (c) two swirls data sets. Blue (red) color indicates small (large) alignment errors.*

rection. From Section 3.1, we can see that surfaces seeded along the binormal direction reveal more interesting 3D shapes than the surfaces seeded along the normal direction. It is easy to verified that a stream surface aligned with the binormal direction everywhere maximizes the average squared normal curvature. For this kind of surfaces, the normal $\mathbf{n_s}$ of the surface at a point p will be perpendicular to the binormal vector $\mathbf{b}$ and flow direction $\mathbf{v}$ at p. Since $\mathbf{Jv}(=\kappa\mathbf{n})$ is also perpendicular to both $\mathbf{b}$ and $\mathbf{v}$ by definition, $\mathbf{n_s}$ and $\mathbf{Jv}$ are parallel, which maximizes the dot product of $\mathbf{n_s}$ and $\mathbf{Jv}$. On the other hand, a stream surface aligned with the normal direction minimizes the mean normal curvature. Although seeding along the binormal direction does not guarantee that the entire surface will align well with the binormal field, our experiment in Section 3.1 shows that it yields a reasonably good approximation. In Figure 10, we show the alignment errors measured on three stream surfaces. The error at a surface point is measured by the angle between $\mathbf{n_s}$ and $\mathbf{n}$. For simpler flow patterns, the timelines mostly follow the binormal direction at all time steps, as shown in (a) and (b). For the more complicated flow pattern shown in (c), although we can observe alignment errors on the boundaries of the surface, the overall error is still acceptable as most of the surface is colored in blue.

In the future, we would like to use the seeding curve following the binormal direction as a starting point to seek an optimal seeding curve. Searching the seeding rakes with different orientations may produce better results than using the binormal direction. However, it is far from optimal as the search space is quite limited. For example, the curvy surfaces in Figure 10 (a) can be hardly captured by straight seeding rakes. We propose to adopt an algorithm based on simulated annealing to gradually optimize the seeding curve. In each iteration of the optimization algorithm, we may randomly move a seed to update the seeding curve and estimate the quality of the generated surface. The new seeding curve is accepted only if the generated surface has better quality than the surface generated by the previous seeding curve. Note that we only need to consider the curves that are perpendicular to the flow everywhere, since moving a seed along the flow direction will not change the generated stream surface. Therefore, a seed can be restricted to move in the plane perpendicular to the flow direction without sacrificing the search space of all possible stream surfaces.

Quality Estimation. The optimization of seeding curves requires the quality of seeding curves to be evaluated efficiently, especially for an interactive approach. We would like to develop a parallel scheme to estimate the quality of the seeding curve without generating the surface. We can first trace a streamline from each seed along the seeding curve and check for the diverging flow by measuring the widths of the ribbons between any two consecutive streamlines. If the maximum width of a ribbon exceeds some threshold, the diverging flow exists and we place additional seeds to fill the gap; otherwise, the ribbon can be considered as part of the surface and used for quality evaluation. In order to avoid excessively diverging the flow, we determine the number of seeds to be placed by the maximum width of the ribbon. Tracing the streamlines and computing the maximum width for each stream ribbon can be performed in parallel, so that the performance is independent of the number of seeds. We repeat this procedure for the new seeds in multiple passes to ensure that the surface quality in the diverging case can be estimated more accurately. The total time cost of this parallel scheme mainly depends on the number of passes taken. With more seeds added for each divergence case, we expect to achieve a reasonably accurate measure within a few passes. We will compare the quality estimation results without and with generating the surface to evaluate the estimation accuracy.

Interaction and Rendering. Currently, once the stream surfaces are confirmed, users are not able to further adjust them. In the future, we would like to enable post-editing so that users can polish the generated stream surfaces. For example, users may simply drag the surface and extend it to include certain flow features. Our current hand tool can only extend the surface along the original seeding curve following the binormal direction. If a flow feature is not passed by the streamlines generated from the extended seeding curve, the surface will not cover the flow feature. An ideal approach should identify a smooth seeding curve that covers both the intended flow feature and the flow patterns of the original surface, if possible. Users may also use the eraser tool to remove a part of stream surface that covers less interesting flow patterns.

Moreover, it may be worthy to investigate surface rendering to enhance the perception of flow directions in complex regions. Many layers of surface can stack in those regions. When the opacity is high, the inner layers are occluded; but when the opacity is low, the flow pattern can be hardly perceived. The opacity based on normal variation considers only surface shape and viewing direction, but not visual occlusion. Our attribute-based coloring can only reduce the opacity for all regions sharing similar values, which does not take occlusion into consideration either. We will design a rendering scheme that can adjust the opacity of surface patches according to their occlusion relationships, allowing users to better perceive flow patterns in complex regions. Additionally, ambient occlusion and other global illumination techniques may be applied to further enhance the rendering results.

7 Conclusions

We have presented a novel sketch-based interface to generate stream surfaces. The interface is designed to follow the commonly used painting metaphor, requiring less learning effort. We provide a suite of tools to users so that they can peel the flow field layer by layer to obtain desired stream surfaces with customized rendering

effects. By following the binormal direction, we efficiently generate surfaces with acceptable quality to support interactive performance. Our approach allows users to obtain customized visualization results that describe the flow field according to their own needs. To the best of our knowledge, our work is the first that leverages user sketching and painting metaphor for semi-automatic stream surface generation.

Acknowledgements

This work was supported in part by the U.S. National Science Foundation through grants IIS-1456763 and IIS-1455886. Special thanks to Dr. Reid Johnson for his narration of the accompanying video and the anonymous reviewers for their insightful comments.

References

BORN, S., WIEBEL, A., FRIEDRICH, J., SCHEUERMANN, G., AND BARTZ, D. 2010. Illustrative stream surfaces. *IEEE Transactions on Visualization and Computer Graphics 16*, 6, 1329–1338.

BOYKOV, Y. Y., AND JOLLY, M.-P. 2001. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *Proceedings of IEEE International Conference on Computer Vision*, vol. 1, 105–112.

BRAMBILLA, A., AND HAUSER, H. 2015. Expressive seeding of multiple stream surfaces for interactive flow exploration. *Computers & Graphics 47*, 123–134.

CAI, W., AND HENG, P.-A. 1997. Principal stream surfaces. In *Proceedings of IEEE Visualization Conference*, 75–81.

CARNECKY, R., FUCHS, R., MEHL, S., JANG, Y., AND PEIKERT, R. 2013. Smart transparency for illustrative visualization of complex flow surfaces. *IEEE Transactions on Visualization and Computer Graphics 19*, 5, 838–851.

DALLMANN, U. 1983. Topological structures of three-dimensional flow separations. Tech. Rep. 221-82 A 07, Deutsche Forschungs- und Versuchsanstalt fuer Luft- und Raumfahrt.

EDMUNDS, M., LARAMEE, R. S., MALKI, R., MASTERS, I., CROFT, N., CHEN, G., AND ZHANG, E. 2012. Automatic stream surface seeding: A feature centered approach. *Computer Graphics Forum 31*, 3, 1095–1104.

EDMUNDS, M., MCLOUGHLIN, T., LARAMEE, R. S., CHEN, G., ZHANG, E., AND MAX, N. 2012. Advanced, automatic stream surface seeding and filtering. In *Proceedings of Theory and Practice of Computer Graphics*, 53–60.

GARTH, C., TRICOCHE, X., SALZBRUNN, T., BOBACH, T., AND SCHEUERMANN, G. 2004. Surface techniques for vortex visualization. In *Proceedings of Eurographics - IEEE TCVG Symposium on Visualization*, 155–164.

GARTH, C., KRISHNAN, H., TRICOCHE, X., BOBACH, T., AND JOY, K. I. 2008. Generation of accurate integral surfaces in time-dependent vector fields. *IEEE Transactions on Visualization and Computer Graphics 14*, 6, 1404–1411.

GUO, H., MAO, N., AND YUAN, X. 2011. WYSIWYG (what you see is what you get) volume visualization. *IEEE Transactions on Visualization and Computer Graphics 17*, 12, 2106–2114.

HULTQUIST, J. P. M. 1992. Constructing stream surfaces in steady 3D vector fields. In *Proceedings of IEEE Visualization Conference*, 171–178.

HUMMEL, M., GARTH, C., HAMANN, B., HAGEN, H., AND JOY, K. I. 2010. IRIS: Illustrative rendering for integral surfaces. *IEEE Transactions on Visualization and Computer Graphics 16*, 6, 1319–1328.

KLEIN, T., GUÉNIAT, F., PASTUR, L., VERNIER, F., AND ISENBERG, T. 2012. A design study of direct-touch interaction for exploratory 3D scientific visualization. *Computer Graphics Forum 31*, 3, 1225–1234.

LOWE, D. G. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2, 91–110.

MARTINEZ ESTURO, J., SCHULZE, M., RÖSSL, C., AND THEISEL, H. 2013. Global selection of stream surfaces. *Computer Graphics Forum 32*, 2, 113–122.

MCLOUGHLIN, T., LARAMEE, R. S., AND ZHANG, E. 2009. Easy integral surfaces: A fast, quad-based stream and path surface algorithm. In *Proceedings of Computer Graphics International*, 73–82.

SADLO, F., PEIKERT, R., AND PARKINSON, E. 2004. Vorticity based flow analysis and visualization for pelton turbine design optimization. In *Proceedings of IEEE Visualization Conference*, IEEE, 179–186.

SCHAFHITZEL, T., TEJADA, E., WEISKOPF, D., AND ERTL, T. 2007. Point-based stream surfaces and path surfaces. In *Proceedings of Graphics Interface*, 289–296.

SCHEUERMANN, G., BOBACH, T., HAGEN, H., MAHROUS, K., HAMANN, B., JOY, K. I., AND KOLLMANN, W. 2001. A tetrahedra-based stream surface algorithm. In *Proceedings of IEEE Visualization Conference*, 151–158.

SCHNEIDER, D., WIEBEL, A., AND SCHEUERMANN, G. 2009. Smooth stream surfaces of fourth order precision. *Computer Graphics Forum 28*, 3, 871–878.

SCHROEDER, D., AND KEEFE, D. F. 2016. Visualization-by-sketching: An artist's interface for creating multivariate time-varying data visualizations. *IEEE Transactions on Visualization and Computer Graphics 22*, 1, 877–885.

SCHROEDER, D., COFFEY, D., AND KEEFE, D. F. 2010. Drawing with the flow: A sketch-based interface for illustrative visualization of 2D vector fields. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Sketch-Based Interfaces and Modeling*, 49–56.

SCHULZE, M., MARTINEZ ESTURO, J., GÜNTHER, T., RÖSSL, C., SEIDEL, H.-P., WEINKAUF, T., AND THEISEL, H. 2014. Sets of globally optimal stream surfaces for flow visualization. *Computer Graphics Forum 33*, 3, 1–10.

TZENG, F.-Y., LUM, E. B., AND MA, K.-L. 2003. A novel interface for higher-dimensional classification of volume data. In *Proceedings of IEEE Visualization Conference*, 505–512.

VAN WIJK, J. J. 1993. Implicit stream surfaces. In *Proceedings of IEEE Visualization Conference*, 245–252.

WEI, J., WANG, C., YU, H., AND MA, K.-L. 2010. A sketch-based interface for classifying and visualizing vector fields. In *Proceedings of IEEE Pacific Visualization Symposium*, 129–136.