Semi-Automatic Generation of Stream Surfaces via Sketching

Jun Tao, *Member, IEEE,* and Chaoli Wang, *Senior Member, IEEE*

Abstract—We present a semi-automatic approach for stream surface generation. Our approach is based on the conjecture that good seeding curves can be inferred from a set of streamlines. Given a set of densely traced streamlines over the flow field, we design a sketch-based interface that allows users to describe their perceived flow patterns through drawing simple strokes directly on top of the streamline visualization results. 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. Furthermore, we propose an optimization scheme to identify the optimal seeding curve in the neighborhood of an original seeding curve based on surface quality measures. To support interactive optimization, we design a parallel surface quality estimation strategy that estimates the quality of a seeding curve without generating the surface. 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.

Index Terms—Flow visualization, sketch-based interface, human perception, seeding curves, stream surfaces.

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 [7], [12].

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 [21].

Inspired by the concept of "structure from motion", which reconstructs 3D structures from a set of 2D images [17], 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 a continuous impression of the underlying flow over the domain, thus providing more immediate hints to searching for good seeding curves. Our goal is to leverage human perception to generate stream surfaces that are picked, adjusted, and verified by users, which ensures that the resulting surfaces will indeed convey the information perceived by users. 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 [2].

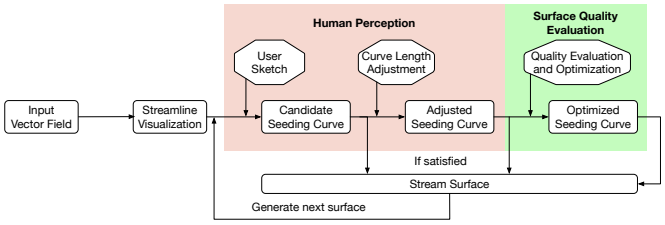


Fig. 1: The overall workflow that uses our sketch-based interface to generate stream surfaces in a semi-automatic manner. The pink part of the workflow involves user interaction while the green part is fully automatic.

In this extended version of our ACM SA16VIS paper [31], we present a sketch-based interface that allows users to generate desired stream surfaces on top of a set of densely traced streamlines. We sketch a typical workflow in Figure 1. Starting from the streamline visualization results, users can simply draw a 2D stroke directly on the streamlines to indicate the favored seeding curve. We derive a binormal field from the flow field and identity a 3D candidate 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

• *J. Tao and C. Wang are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556. E-mail: {jtao1, chaoli.wang}@nd.edu.*

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 adjust the length 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 apply an additional optimization step to improve the seeding curve based on our surface quality estimation. If the seeding curve is satisfactory, they can generate the stream surface at any step during this procedure; otherwise, they can abandon the current seeding curve by drawing another stroke. This procedure applies human perception to position a seeding curve and verifies the generated surface to ensure that the visualization results can be perceived in the desired way. Users can 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.

The contributions of our work are the following: First, we design a novel interface that allows users to generate desired stream surfaces by sketching on top of a set of densely traced streamlines. Compared with trial-and-error manual stream surface seeding, our approach provides useful indications for users to predict the shapes of generated surfaces. Compared with the fully automatic surface selection, our approach is more flexible by allowing users to generate stream surfaces according to their own needs. Second, our interface leverages a painting metaphor with four tools (pencil, hand, eraser, and brush) which most users are familiar with. This reduces the learning curve and allows the stream surfaces to be generated and rendered in the "what-you-see-is-what-you-get" manner. Third, we propose a parallel scheme to estimate the surface quality from a seeding curve without actually generating the surface. Based on this quality estimation, we further introduce a real-time surface optimization approach to improve the seeding curve in its local neighborhood.

2 Related Work

For almost thirty years, flow visualization has been a central topic in scientific visualization. We refer readers to the surveys of integration-based [22] and surface-based [8] flow visualization for an overview of related work. In the following, we restrict our attention to stream surfaces and review related work on surface seeding and placement, surface construction and computation, surface rendering, and surface selection. Then, we present related work on sketch-based interface for scientific visualization. We also highlight the differences between our work and prior work.

2.1 Surface-Based Flow Visualization

Existing stream surface generation techniques include extracting stream surfaces as isosurfaces in specifically designed scalar fields [4], [33], creating seeding curves as isolines of scalar fields on domain boundaries [10], and generating seeding curves based on streamline clusters [9] or a seeding structure (e.g., a seeding plane) [3], [24]. Similar to our approach, Sadlo et al. [24] proposed to generate seeding curves guided by a derived vector field (vorticity field). Unlike the binormal direction, which is uniquely determined by the shape of streamlines, vorticity field lines are impacted by many other factors (e.g., velocity magnitude). Therefore, although their approach is effective for vortex analysis, it may not produce surfaces with desired shape properties for general flow patterns.

Hultquist [14] 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 [12], quad-based [23], and point-based [25] algorithms. Other works improve the interpolation using tetrahedral grids [26] or Hermite interpolation [27]. Garth et al. [11] advocated a two-step surface generation. It first generates a skeleton of the integral surface, followed by a well-conditioned triangulation. Machado et al. [18] represented surfaces using a dense set of streamlines, which does not require any triangulation of surfaces.

Surface rendering aims at reducing visual occlusion and clutter and enhancing the depth and spatial perception of flow features and structures. Existing techniques leverage contour lines and half-toning [1], transparency and texturing [15], and illustration buffer [5] to improve and enhance the perception of surfaces. Born et al. [1] leveraged contour lines and half-toning to show the overall surface shape. They demonstrated that flow directions and singularities on the stream surface can be depicted and enhanced by illustrative surface streamlines. Hummel et al. [15] studied how transparency and texturing can be used to convey the shape and directional information. They applied screen-space curvature approximation to enhance integral surface visualization. To enable nonlocal transparency enhancement and create expressive surface rendering, Carnecky et al. [5] proposed the illustration buffer to store a list of all surface layers for each pixel and used a set of operators to process these depth lists to generate the final image.

Only a few works are directly related to stream surface selection [20], [30]. Martinez Esturo et al. [20] pointed out the huge search space of possible stream surfaces in a flow field and 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. [30] extended the work to select a set of globally optimal stream surfaces in an iterative manner. All selected surfaces are mutually distant to convey different flow features while reducing visual occlusion and clutter. Together they optimize global stream surface quality measures. These two solutions 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.

2.2 Sketch-Based Interface for Scientific Visualization

In scientific visualization, sketch-based interface leverages human perception and intuition to achieve customized results and convey essential information. For volume visualization, sketch- and touch-based interface and interaction have been introduced for transfer function design [32], WYSIWYG volume visualization [13], and

Visualization-by-Sketching for time-varying multivariate data visualization [29]. Tzeng et al. [32] introduced a sketch interface for transfer function specification. The interface allows users to paint directly on sample slices of the volume to specify the materials that they want and do not want to see. Such information is fed into an artificial neural network for designing a high-dimensional classification function for volume visualization. Guo et al. [13] developed a set of tools to enable direct manipulation of color, transparency, contrast, brightness, and other optical properties by brushing a few strokes on top of the volume rendering image. By matching the sparse sketching input with the clustered features in both image space and volume space, they were able to identify the targeted volumetric features. Both works [13], [32] provide users with intuitive and flexible interaction, without resorting to the traditional approach of tuning transfer function parameters via sliders, buttons, or other widgets. Schroeder et al. [29] presented Visualization-by-Sketching, a sketching interface to enhance the role of human creativity in time-varying multivariate data visualization. Designers are allowed to paint directly on top of a digital data canvas, sketch data glyphs, and arrange and blend together multiple layers of animated 2D graphics.

For flow visualization, sketch-based interfaces have been applied to 2D illustrative visualization [28], 3D flow field classification [35], and exploration of scientific data sets [16]. Schroeder et al. [28] designed a sketch-based interface for 2D vector fields that enables users to draw on top of LIC images. Streamlines approximating the strokes are then added into the visualization. It also allows users to cut, extend or completely remove a streamline using sketching. Wei et al. [35] targeted 3D vector fields instead and presented a solution that allows users to sketch a 2D curve for pattern matching in 2D and streamline clustering in 3D. A string matching method is utilized to identify a 3D streamline whose view-dependent 2D projection is most similar to the user's drawing. Then, the identified streamline is used to extract all similar 3D streamlines. Klein et al. [16] presented a design study of direct touch interaction to explore 3D scientific data sets. Users can manipulate a 2D cutting plane on a touch screen, and select endpoints where streamlines are traced in between.

Unlike the sketch-based interfaces presented in [28], [35] 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 [32]. This minimizes the gap between users' perceived flow patterns from the streamlines and the generated stream surfaces. Compared to the seeding rake generation [16], 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.

Compared with our ACM SA16VIS paper [31], this journal version makes a significant extension by introducing 1) the parallel surface quality estimation that efficiently estimates the quality of a seeding curve without generating the actual surface, and 2) the optimization that improves a seeding curve in its local neighborhood based on the surface quality estimation. The extension also includes a quantitative study of the new estimation and optimization step to demonstrate its effectiveness.

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). To avoid ambiguity, we define $\mathbf{t}$ as a unit vector pointing in the same direction as the flow direction $\mathbf{v}$. The directions of $\mathbf{n}$ and $\mathbf{b}$ are decided accordingly. 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 [23].

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 defined by the normal $\mathbf{n}$ and binormal $\mathbf{b}$ vectors. P_n is perpendicular to the flow direction (i.e., the tangent vector $\mathbf{t}$). 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. In Figure 2 (b), we illustrate the seeding curves along the normal direction $\mathbf{n}$ and binormal direction $\mathbf{b}$ in red and green, respectively. The black curves are three planar streamline segments l_1, l_2, l_3 residing in the plane P_l. We can see that both $\mathbf{n}$ and $\mathbf{b}$ are perpendicular to the streamline segments, but $\mathbf{n}$ lies in the same plane of P_l as the streamline segments while $\mathbf{b}$ is perpendicular to P_l. Therefore, seeding along $\mathbf{n}$ (red curves) will create a planar surface, which only depicts the flow direction on the boundary of the surface (i.e., l_1 and l_3). The flow direction inside the surface, such as the shape of l_2, can only be implied from the *boundary* of the surface. On the contrary, seeding along $\mathbf{b}$ is more desirable, since it leads to a surface with a more interesting 3D shape, allowing the flow direction to be perceived from the *shape* of the surface.

Figure 2 (d) and (e) show two stream surfaces traced from two seeding curves following the 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 proposed by Martinez Esturo et al. [20] to evaluate the quality of a surface. They measured the normal curvature of the surface at a point p as

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

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

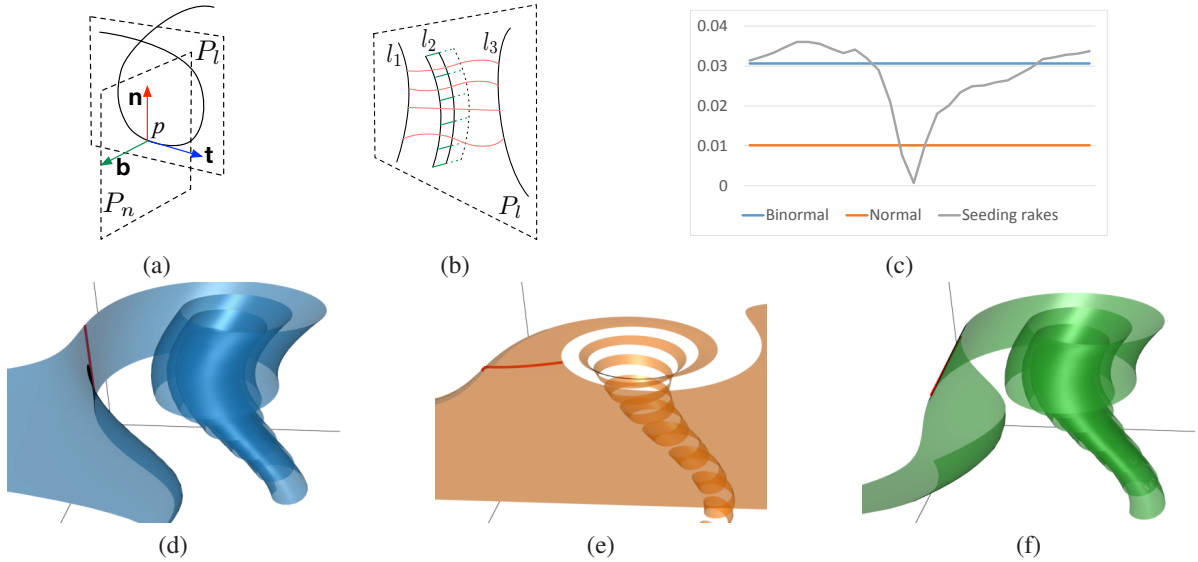


Fig. 2: (a) the tangent **t**, normal **n**, and binormal **b** vectors at a point p on a streamline. (b) seeding along **n** (red curves) or **b** (green curves). (c) quality measure using the average squared normal curvature. The horizontal axis corresponds to equally sampled seeding rake directions, starting from the binormal direction. (d) to (f) show stream surfaces seeded along the binormal direction, the normal direction, and the best straight seeding rake, respectively, for the tornado data set.

same direction as **t** with additional magnitude information. For a 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. 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 (c), starting from the rake following the binormal direction. The average squared normal curvature of the 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 (d) and (f), 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

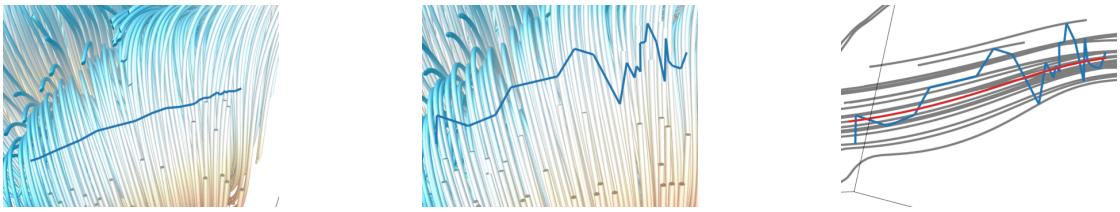
The seeding curve approximation is performed as follows. First, we map the points on the 2D user-drawn stroke back to the 3D space. For each point p, we identify the streamline that is first hit at p, and use the hit point as the mapped 3D point. Figure 3 (a) shows a user-drawn stroke in the original 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 trace a curve 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 (MCP) 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

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

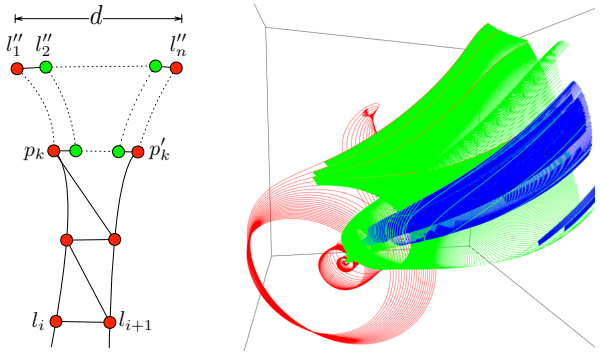
where $d(p_i, q_j)$ is the Euclidean distance between p_i and q_j. Figure 3 (c) demonstrates one example with this setting. One gray curve is generated at each point on the blue 3D stroke, and the closest curve is the one highlighted in red. We can see that the depth randomness of streamlines is canceled 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



(a) (b) (c)

Fig. 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.



(a) (b)

Fig. 4: Quality estimation based on streamlines. (a) shows the case where additional streamlines are added for the diverging flow. (b) shows the streamlines added in three passes using the five critical points data set. The red, green, and blue streamlines are added in the first, second, and third passes, respectively.

points identified is the seeding curve. In Figure 3 (c), the red curve segment is the seeding curve we generate for this example.

3.3 Surface Quality Estimation

We estimate the quality of a seeding curve by predicting how well its corresponding stream surface aligns with the flow. The estimation is based on a set of streamlines traced from the seeding curve. Since we do not require the exact surface for quality estimation, our solution is more suitable for supporting runtime interaction. We perform quality estimation for every surface strip formed between two consecutive streamlines. The quality of the entire surface is obtained by aggregating the quality of all strips.

Figure 4 (a) shows one strip between two consecutive streamlines l_i and l_{i+1}. We apply the greedy approach described by Hultquist [14] to triangulate this strip so that the local shape of the surface can be better represented using the triangles. The quality of the surface is evaluated at each triangle and aggregated over the entire surface weighted by the area of triangles. We consider two terms used by Martinez Esturo et al. [20] to measure the quality, namely, alignment error and normal curvature. Both terms can be evaluated locally. The alignment error evaluates how well the principal directions of the surface align with the flow directions so that the flow direction on the surface can be captured by human perception. The normal curvature is used to avoid generating trivial surfaces where less information can be perceived.

However, if the flow diverges, this quality measure between neighboring streamlines may be inaccurate as the triangles could be highly skewed. In Figure 4 (a), the left parts of the two streamlines align well with each other. But the right parts diverge as the two streamlines go into different directions. The surface between them cannot be faithfully represented by a single strip anymore. To tackle the divergence case, we keep track of the distance between the closest point pair along the two streamlines l_i and l_{i+1}. When this distance increases to twice of the distance between two consecutive seeds d_s, we terminate the computation of quality for the two streamlines and trace multiple streamlines to fill the gap. The number of streamlines to add is estimated by $n = d/d_s$, where d is the distance between the endpoints of the two streamlines l_i and l_{i+1}. In Figure 4 (a), we terminate quality evaluation at points p_k and p'_k and add n new streamlines (i.e., $l''_1, l''_2 \ldots l''_n$). The surface quality is then estimated using the new streamlines.

The added streamlines could be overly dense, which only increases the number of samples but does not change the quality significantly. This is because the result is weighted by the local area of points, and the total area does not change much as the density of streamlines varies. As such, we allow up to three levels of divergence cases in our implementation. In Figure 4 (b), we can see that the surface is well captured with the additional streamlines added in the second and third passes.

The time complexity of computing the quality between two streamlines is linear with respect to the number of points on the streamlines given the triangulation since it does not require testing the pairwise distance among points. Without considering the divergence case, the alignment error between different streamline pairs is independent of each other. This means that the computation can be easily parallelized to reduce the evaluation time. The use of overly dense streamlines for each divergence case increases the number of streamlines. However, it reduces the level of divergence cases, which is desired for parallel computation. With the maximum level of divergence cases set to three, our quality estimation for the entire surface can be completed in three passes. In each pass, the alignment error between streamlines at the same level is computed. Therefore, if computed in parallel, the quality estimation for the entire surface is still linear with respect to the number of points on a single streamline. In contrast, the performance of quality measures that directly evaluate stream surfaces is often constrained by surface integration [20].

3.4 Seeding Curve Optimization

Seeding curve optimization is an optional step we consider for improving the surface quality. The original seeding curve that approximates a user's sketch follows the binormal direction to generate the surface. However, even if the binormal vector field is used to guide surface generation, it does not guarantee that the resulting stream surface aligns well with the binormal vector field globally. At this stage, we search for other seeding curves that

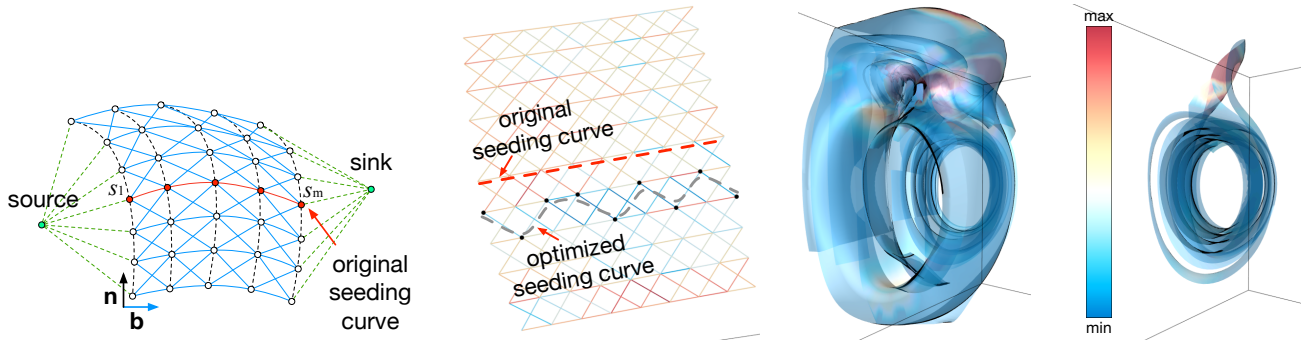


Fig. 5: Seeding curve optimization. (a) shows an example of a grid generated from the red seeding curve. (b) shows a grid using the two swirls data set. (c) and (d) show the stream surfaces generated from the original and optimized seeding curves based on (b), respectively.

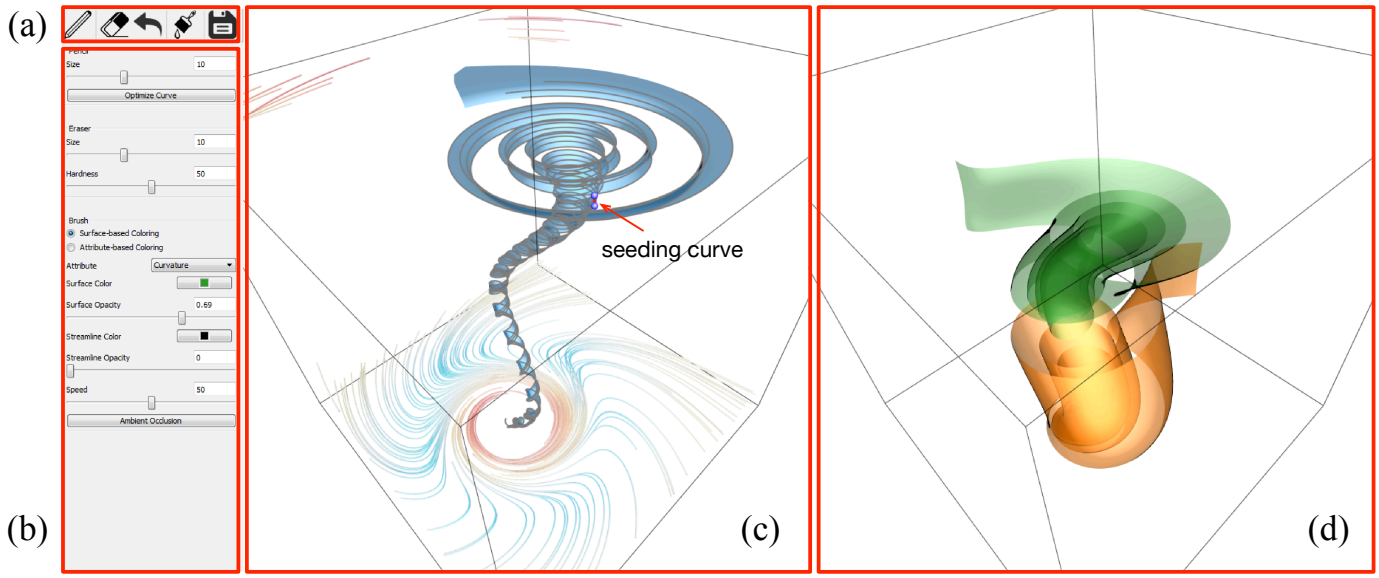


Fig. 6: Our sketch-based interface for stream surface generation. (a) to (d) show the panel of tools, the panel of tool parameters, the streamline widget, and the stream surface widget, respectively. In (a), the five tool icons displayed from left to right are pencil, eraser, undo, brush, and save, respectively. Users sketch to generate a new stream surface in (c). Once confirmed, the newly generated surface will be moved to (d) for visualization.

produce stream surfaces covering similar flow patterns with better quality. This procedure is outlined in Algorithm 1.

The search space is a grid G of candidate seeds S_G by moving the seeds on the original seeding curve along the normal direction, as shown in Figure 5 (a). The original seeding curve and the candidate seeds are displayed as the red curve and the red circles, respectively. For each seed s_i on the seeding curve $S = \{s_1, \ldots, s_m\}$, we trace s_i along the normal direction up to k steps forward and k steps backward. The width of the grid is constrained by the number of steps k so that the optimized seeding curve still cover similar regions. Note that such a search space is sufficient since the grid is likely to be perpendicular to the flow direction as it aligns with the binormal and normal directions. Therefore, for any seeding curve, an equivalent seeding curve that is not on the grid can be found by moving the seeds along the flow direction.

Compared with the surface optimization on a regular 3D grid proposed by Martinez Esturo et al. [20], our 2D grid is more efficient since it has fewer grid points. However, our 2D grid may not be fully expanded when critical points exist in the normal and binormal fields. The critical points in these two fields satisfy $\mathbf{n} = \mathbf{Jv} = \mathbf{0}$, indicating less interesting laminar flow if $\mathbf{v} \neq \mathbf{0}$. Therefore, it is unlikely to further improve the quality of a surface at the critical points in these two fields. For efficiency, we simply stop expanding the grid at these critical points.

The neighboring candidate seeds are connected to form the edges E_G of the grid. In Figure 5 (a), the connections among candidate seeds are shown by the solid lines. Note that to avoid sharp turns on the seeding curve, we do not connect candidate seeds along the normal direction.

We formulate this optimization problem as finding the shortest path between any two candidate seeds $s_{1,i}$ and $s_{m,j}$ spawned by the first seed s_1 and last seed s_m on the original curve, so that the optimized seeding curve roughly follows a similar direction and covers a similar range. As shown in Figure 5 (a), we create two dummy nodes as source s_0 and sink s_{m+1} (green circles), which connect to candidates spawned by the first and last seeds, respectively, with weights of zero (green dashed lines). The shortest path between the source and sink is then identified using Dijkstra's algorithm.

Algorithm 1: Seeding curve optimization

input : A seeding curve $S = \{s_1, \ldots, s_m\}$
output: An Optimized seeding curve S'

$G =< S_G, E_G >=< \emptyset, \emptyset >$
foreach $s_i \in S$ **do**
  for $j = -k, \ldots, k$ **do**
    $s_{i,j} \leftarrow$ trace s_i in the normal vector field by j steps
    $V_G \leftarrow V_G \cup s_{i,j}$
  end
end
foreach $s_{i,j}, s_{r,l} \in V_G$ **do**
  if $|i-r| = 1, |j-l| \leq 1$ **then**
    $E_G \leftarrow E_G \cup < s_{i,j}, s_{r,l} >$
  end
end
create two dummy grid points s_0 and s_{m+1}
foreach $s_{1,j} \in G$ **do**
  $E_G \leftarrow E_G \cup < s_0, s_{1,j} >$
end
foreach $s_{m,j} \in G$ **do**
  $E_G \leftarrow E_G \cup < s_{m,j}, s_{m+1} >$
end
evaluate the quality of $< s_{i,j}, s_{r,l} > \in E_G$ in parallel
find the shortest path $s_0 \rightarrow S' \rightarrow s_{m+1}$ on the grid G

Figure 5 (b) shows a grid generated from a seeding curve using the two swirls data set with $k = 5$. The edges are colored by the estimated alignment error of the corresponding stream surface strip. The original seeding curve is the one in the middle with edges of medium or small alignment errors (as indicated by the red dashed curve). The selected seeds on the shortest path are displayed as black dots forming the optimized seeding curve (as indicated by the gray dashed curve). We can see that the optimized seeding curve contains mostly blue segments with small alignment errors. Figure 5 (c) and (d) show the stream surfaces generated from the original and optimized seeding curves, respectively. The stream surfaces are colored by the local alignment error at each vertex using the color map in (d). This color map is used throughout the paper to indicate attribute values on surfaces or velocity magnitude on streamlines. The surface generated from the optimized seeding curve reduces the total alignment error by avoiding most regions with large alignment errors while capturing a similar flow pattern.

4 Interface and Interaction Design

As shown in Figure 6, 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.

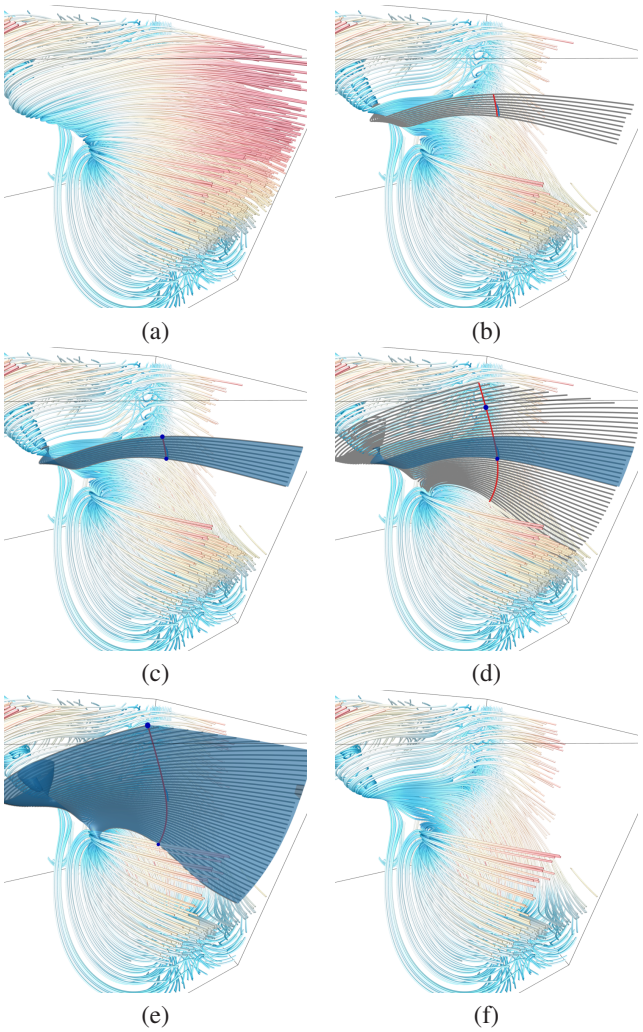
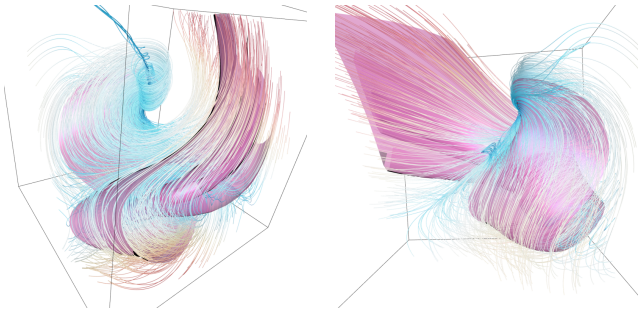


Fig. 7: Generating a single stream surface. (a) shows the original streamlines. (b) shows a user-drawn stroke in blue, the corresponding seeding curve in red, and guiding streamlines in gray. (c) shows the stream surface generated from the seeding curve. (d) shows the extended seeding curve and the corresponding guiding streamlines. (e) shows the stream surface generated from the extended seeding curve. (f) shows the remaining streamlines after removing the streamlines that are captured by the newly generated stream surface.

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 7. We start with a pool of 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



(a) (b)

Fig. 8: 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.

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 to 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 8, 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 depth. 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 depth 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, ambient occlusion, 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 together with the use of ambient occlusion. 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. This 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 et al. [15], 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 $[0,1]$. A large γ value emphasizes regions with high normal variation. We further restrict the transparency value to $[\alpha_{\min}, \alpha_{\max}]$ by linear interpolation $(1-\alpha_v)\alpha_{\min} + \alpha_v \alpha_{\max}$.

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

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

Using the brush tool, users can simply click on a surface to apply the current parameter values of the brush.

Attribute-Based Coloring. This 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. 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

				timing (in seconds)				quality (%)		
data set	dimension	avg. # p/l	SR	curve fitting	surface integral	line removal	optimize	estimation error	AE improve	NC improve
Bénard	$128 \times 32 \times 64$	288.4	1	0.21	0.50	0.11	0.77	15.1	5.6	8.4
crayfish	$322 \times 162 \times 119$	288.7	15	0.23	4.45	0.97	1.63	19.1	9.0	4.1
electron	$64 \times 64 \times 64$	58.0	1	0.26	0.10	0.10	0.16	4.5	13.0	1.0
five CPs	$51 \times 51 \times 51$	116.2	1	0.22	0.15	0.15	0.16	3.6	20.5	5.9
sq. cylinder	$192 \times 64 \times 48$	195.9	1	0.69	2.37	0.41	0.67	26.2	32.0	3.6
tornado	$64 \times 64 \times 64$	292.4	1	0.25	0.26	0.14	0.15	13.1	20.1	11.6
two swirls	$64 \times 64 \times 64$	918.2	10	0.79	2.24	0.73	1.54	21.4	35.6	10.4

TABLE 1: Timing and quality results, and parameters for each data set. We used 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. "SR" is sampling rate, "AE" is alignment error, and "NC" is normal curvature.

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.

5 Results

5.1 Performance, Parameters, and Quality Estimation

We evaluated the performance of our approach using the data sets listed in Table 1. From top to bottom, these data sets are: the liquid flow between two parallel planes [36], the heat flow around a cooking crayfish [19], a flow field containing three electrodes [37], a synthesized flow field consisting of five critical points [38], the flow around a confined square cylinder [34], a procedurally generated tornado [6], and swirls resulting from wake vortices [19]. 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 video memory.

Performance and Parameters. For most data sets, we randomly traced 3000 streamlines to fill the entire domain. For the crayfish and two swirls data sets, since the streamlines are relatively long with repeated patterns, we used fewer streamlines for better speed performance. Generally, the streamlines are sufficiently dense if most of the points on a user sketch will fall onto the outermost layer of the streamlines. 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, the timing and quality results were collected in three runs. During each run, we generated as many surfaces as needed, until most streamlines were removed and the flow features were captured by the surfaces. For different runs, we might generate different sets of surfaces to collect more complete results. 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 seeding curve optimization time includes the time to estimate surface strips between every pair of candidate seeds and find the shortest path on the grid. The streamline tracing and removal, and surface quality estimation were performed in the GPU, and the other steps were performed in the CPU. The guiding streamlines contain much fewer points than the final surface, especially when the flow is complex. Therefore, removing streamlines according to the guiding streamlines is less costly.

In our experiment, the average time to fit a seeding curve and generate the guiding streamlines is less than one second for all data sets. The surface integral may be costly for more complicated data sets since the diverging and swirling flow patterns will result in a large surface, which can take several seconds. But in general, the system is interactive as the surface integral and streamline removal steps will only be performed when users are satisfied with the guiding streamlines. Our parallel surface quality estimation provides significant speedup to support real-time optimization. Given a seeding curve with m seeds, the optimization will need to evaluate the quality of $(6k+1) \times (m-1)$ surface strips and identify the shortest path, where $k = 5$ is the number of candidate seeds produced by an original seed in both forward and backward directions. This leads to around 31 times of computation when the surface integral is required in the optimization. With our parallel estimation, the optimization takes less than one second for most of the data sets and it is usually even faster than surface integral.

Quality Estimation. To measure the accuracy of our quality estimation, we compare the estimated alignment error of each seeding curve with the one measured on the actual stream surfaces. The estimated error is small for more regular flow fields (less than 5% for the electron and five critical points data sets) and relatively large for more complex ones (26.2% and 21.4% for the square cylinder and two swirls data sets, respectively). In our experiment, we find that the estimation usually covers a larger area than the actual surface when the estimated error is large. This is likely due to different termination criteria of the streamline integral for estimation and the actual surface integral. Our surface integral implements the approach proposed by McLoughlin et al. [23], which produces smooth timelines by adjusting the integration step sizes for different seeds when flow rotates. Therefore, the length of each streamline on the resulting surface depends on others. In contrast, the lengths of streamlines generated in the estimation are not constrained by others, and the estimation usually covers a more complete area.

Generally, the estimation is accurate enough to guide the optimization. For each seeding curve, we produce one optimized curve by *minimizing* the alignment error and another by *maximizing* the normal curvature. We then compare the surfaces generated from the optimized ones with the original ones. In terms of the alignment error, we find that more reduction can be obtained through the optimization for complex data sets (35.6% and 32% less alignment error for the two swirls and square cylinder data sets, respectively). Understandably, if the flow is simple, seeding

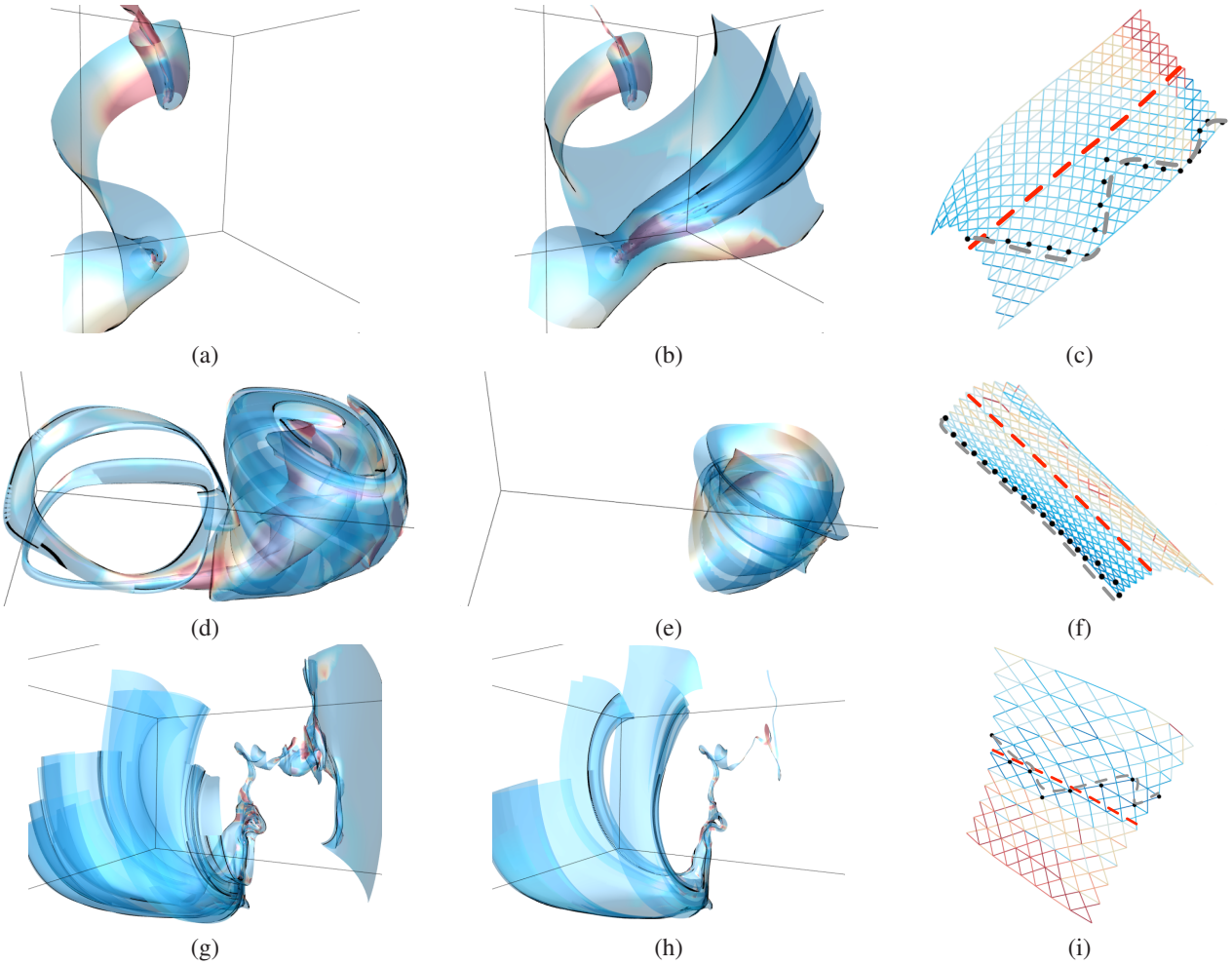


Fig. 9: Minimizing the alignment error for the five critical points, Bénard flow, and crayfish data sets. The first and second columns show the surfaces before and after optimization, respectively. The third column shows the grids of candidate seeds. The black dots indicate the selected seeds. The red and gray dashed curves indicate the original and optimized seeding curve, respectively. Surfaces and grids are colored by the alignment error.

curves in the local neighborhood of the original one are likely to produce similar surfaces; only with more complicated flow patterns, surface strips produced on the grid of candidate seeds are more diverse, leaving more room for potential improvement. The only exception is the crayfish data set, which is complex but our optimization only manages to reduce 9% of alignment error. Figure 9 shows three original surfaces and their corresponding optimized ones using three different data sets. From the grids of candidate seeds, we can see that the optimized seeding curves contain mostly blue segments, indicating that the alignment error is small. For all three cases, the error reduction is more than 30%. By comparing the optimized surfaces with the original ones, we find that they not only tilt to better align with the flow. More importantly, the optimized surfaces either avoid the regions with large alignment error or include more regions with small alignment error to obtain such reduction. In terms of the normal curvature, we find that usually less improvement can be achieved through optimization as the increase of normal curvature is less than 10% for most of the data sets. This indicates that the original seeding curves already produce surfaces that align well with the binormal direction. Therefore, there is little room to improve over the original seeding curves.

Figure 10 demonstrates the optimization results of maximizing the average squared normal curvature (top row) and minimizing a linear combination of the alignment error and the average squared normal curvature (bottom row). The same seeding curve is used as the seeding curve in the bottom row of Figure 9. By maximizing the normal curvature, stream surfaces with nontrivial shapes are preferred. In Figure 10 (a), we can see that the surface mostly resides in the region of more complicated flow patterns. Note that the high normal curvature regions (red) are preferred. Then, we investigate the optimization using both the alignment error ε and the normal curvature κ. We empirically combine the two attributes as $\varepsilon - 10\kappa$. We use a negative weight so that higher κ is preferred during the minimization of the combined term. The scaling factor 10 is used to balance the impact of the two quality measures. We find that the surface in Figure 10 (c) is similar to the corresponding surface in Figure 9 (a). But the region of simpler flow pattern is covered less since the normal curvature is involved.

Since we focus on evaluating how well our optimization can locally adjust an existing seeding curve for better surface quality, we do not further investigate the impact of different quality measures and how these measures should be combined. Our quality estimation and seeding curve optimization should work with any other quality measures that can be locally evaluated. To enhance the perceptual quality, we may need further criteria that

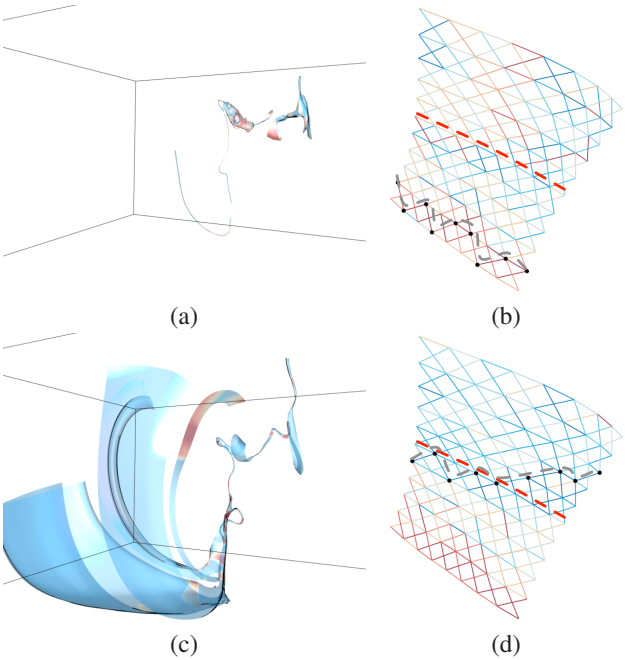


Fig. 10: Surface optimization results using the crayfish data set. (a) and (b) show respectively, the optimized surface and the grid of candidate seeds that maximizes the normal curvature. (c) and (d) show respectively, the optimized surface and the grid of candidate seeds that minimizes a combination of the alignment error and the normal curvature. Surfaces and grids are colored by the respective quality measure.

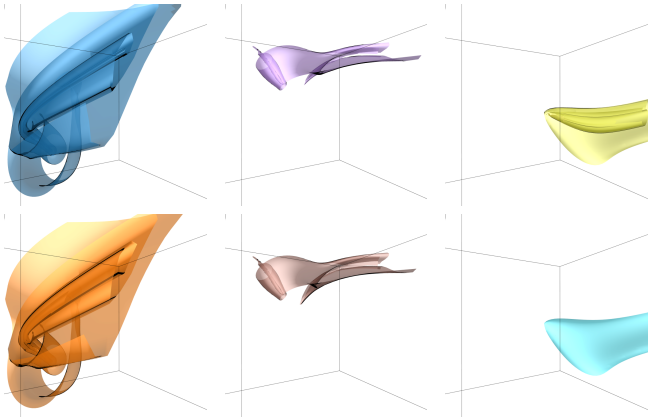


Fig. 11: Stream surfaces generated by tracing seeding curves in a precomputed binormal vector field (top row) and by computing the binormal directions in the original vector field in real time (bottom row). The five critical points data set is used.

consider the information conveyed by the surface or the shared information between a surface and previously generated ones to avoid the overly simplified surface (e.g., Figure 9 (d)) or surfaces with redundant information.

Binormal Computation. In our implementation, we trace the seeding curves in a precomputed binormal vector field (PRE). An alternative is to compute the binormal directions in the original vector field in real time (RT), which may be more accurate but would be more costly. In Figure 11, we can see that the stream surfaces generated using the two methods are very similar. The most significant difference is found between the two surfaces in (c), where PRE captures the spiral at the center, but RT does not.

data set	average quality		difference (%)		MCP
	PRE	RT	quality	area	distance
Bénard	0.69	0.67	5.18	16.3	1.48
crayfish	0.59	0.59	3.76	19.4	2.02
electron	0.96	0.96	0.06	1.6	0.27
five CPs	0.69	0.68	4.63	16.6	2.49
sq. cylinder	0.69	0.67	4.76	15.5	0.87
tornado	0.81	0.82	1.70	5.7	1.02
two swirls	0.68	0.69	2.99	10.5	1.03

TABLE 2: The performance difference of using a precomputed binormal vector field and computing the binormal directions in real time. "average quality" shows the average quality evaluated by the alignment error. "MCP distance" shows the MCP distance between the corresponding stream surfaces.

We use the MCP distance (Equation (3)) to quantify the difference. The distances between surfaces in (a), (b), and (c) are 0.71, 0.86, and 2.05, respectively.

For a more complete study, we randomly generate 20 points in each data set and trace 20 seeding curves along the binormal direction using each method. Each seeding curve contains 20 seeds. The results are shown in Table 2. For most of the data sets, we can see that the distances are smaller than 2. The smallest average distance (0.27) is found using the electron data set with less diverging flow patterns. The five critical points data set has the largest average distance (2.49) due to the existence of a large distance (17.75) between two corresponding surfaces. The alignment errors for most of the data sets are similar, with the largest difference of 5.18% for the Bénard data set. The average alignment errors are similar as well, and none of the two methods is a clear winner.

In general, we find that the seeding curves generated using the two methods are mostly identical, but their differences may get larger with the presence of the diverging flow. In our experiment, we find that the differences between stream surfaces generated by randomly selected seeding curves are usually larger than the differences between stream surfaces generated by user sketches. This is because users tend to sketch on stable flow patterns rather than turbulent ones.

5.2 Visualization Results

Five Critical Points. The six stream surfaces we generate from the five critical points data set are shown in Figure 12. 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. Therefore, we add streamlines to enhance the perception of these critical points. The saddle between the green and red

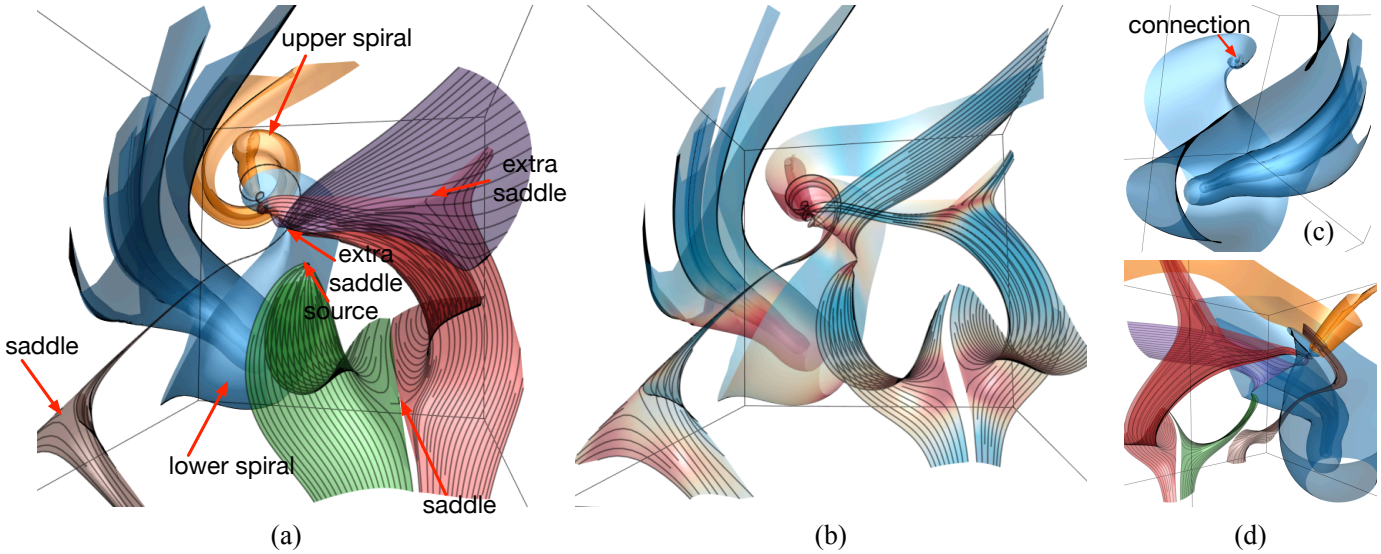


Fig. 12: (a) and (b) show six stream surfaces of the five critical points data set with surface-based coloring and attribute-based coloring, respectively. (c) shows the single blue stream surface. (d) shows the surfaces as seen from the opposite viewpoint of (a).

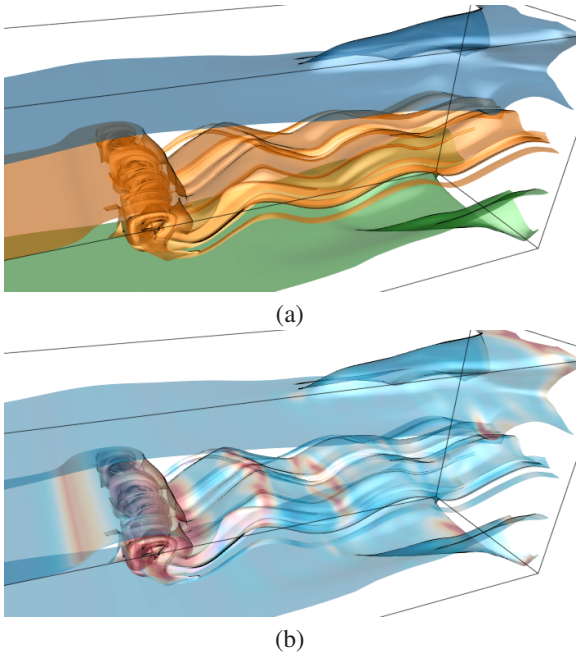


Fig. 13: (a) and (b) show three stream surfaces of the square cylinder data set with surface-based coloring and attribute-based coloring, respectively.

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

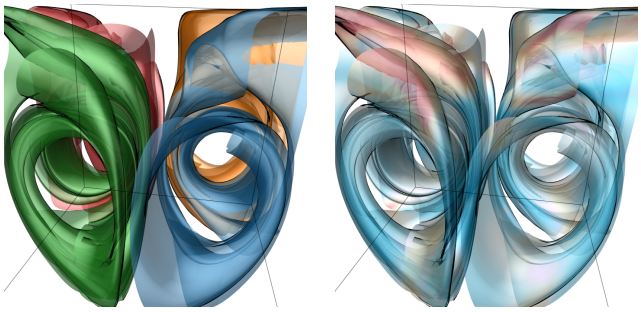
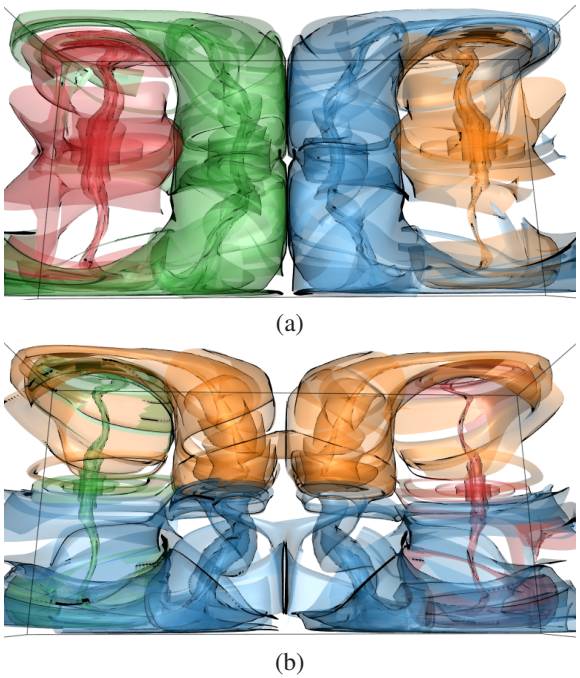


Fig. 14: (a) and (b) show four stream surfaces of the two swirls data set with surface-based coloring and attribute-based coloring, respectively.

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 13 to 16. We generate three surfaces for the square cylinder data set to capture the upper, middle and lower layers of the flow (Figure 13), and four surfaces for the two swirls data set to capture the flow patterns in the front and back halves of each swirl (Figure 14). In Figure 15, we show two sets of surfaces for the Bénard flow data set which demonstrates the flexibility of our approach: Figure 15 (a) resembles Figure 15 in [9] and Figure 15 (b) resembles the corresponding subfigure of Figure 7 in [30]. In Figure 16 (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 16 (b). The flow pattern can be captured by a single surface as well, as shown in Figure 16 (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.

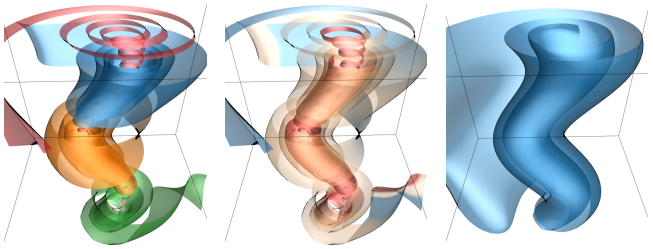


(a) (b)

Fig. 15: (a) and (b) show two different sets of stream surfaces of the Bénard flow data set with surface-based coloring.

5.3 Limitations

Our current work has several limitations. First, the accuracy of our quality estimation still has room for improvement. In Section 5.1, we discuss two scenarios that may lead to inaccurate estimation results: 1) the gaps in the diverging flow that cannot be filled within three passes, and 2) the surface integration method [23] and our estimation may cover different lengths along the streamlines. While we consider the first scenario to be a tradeoff between accuracy and efficiency, the second scenario may be solved by developing a surface integration method that is more consistent with our current estimation strategy. A similar parallel scheme may be used to generate streamlines in several passes and connect the neighboring streamlines to form the surfaces. In this way, the streamlines traced in the estimation stage may better represent the actual surfaces. Certainly, we still need to investigate how to handle the rotation, converging, and diverging flows properly to produce smooth surfaces with this parallel scheme. Second, the performance of our seeding curve optimization relies on the accuracy of the estimation. Although our seeding curve optimization only performs on a relatively small 2D grid, we find that this optimization usually performs well with accurate estimation results. However, when the estimation becomes inaccurate, the optimization may produce less desired results. For example, in Figure 10 (c), the value of combined term is reduced by 60.2% with the optimization, but the quality evaluated on the actual surface is only 8.7% better than the original surface. Third, the optimized surfaces are not guaranteed to capture user intention. For example, the optimized surfaces may be overly simplified (e.g., Figure 9 (d)) to avoid entering regions with a higher alignment error. In this case, they may present flow patterns quite different from the original ones. We may need to consider the information conveyed by the surface or the shared information between a surface and previously generated ones so that the optimized surface conveys similar information in a clearer way.



(a) (b) (c)

Fig. 16: (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.

6 Conclusions and Future Work

We have presented a novel sketch-based interface to generate stream surfaces guided by human perception. 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. We also provide an option to allow users to further improve seeding curves based on surface quality measures. Our approach allows users to obtain customized visualization results that describe the flow field according to their own needs and to verify that the surfaces convey the information they perceive. To the best of our knowledge, our work is the first that leverages user sketching and painting metaphor for semi-automatic stream surface generation.

In the future, we would like to develop a surface integration approach that is more consistent with our quality estimation. We will evaluate how this may improve the accuracy of the estimation and the performance of the seeding curve optimization. We will consider the information conveyed by the optimized surfaces and the original ones to ensure that the optimized ones still capture user intention. We would also like to further investigate surface rendering to enhance the perception of flow directions, especially in complex regions. Specifically, we will evaluate the information shared by the surface patches so that when patches with similar information occlude each other, only one patch will be shown with high opacity. When patches with different information occlude each other, we will study the problem of presenting the mixture of information in a possibly abstract way.

Acknowledgments

This work was supported in part by the U.S. National Science Foundation through grants IIS-1456763 and IIS-1455886. We would like to thank the anonymous reviewers for their insightful comments.

References

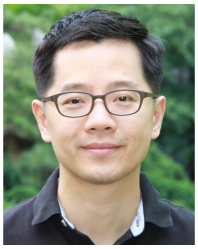
[1] S. Born, A. Wiebel, J. Friedrich, G. Scheuermann, and D. Bartz. Illustrative stream surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1329–1338, 2010.

[2] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *Proceedings of IEEE International Conference on Computer Vision*, pages 105–112, 2001.

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

[4] W. Cai and P.-A. Heng. Principal stream surfaces. In *Proceedings of IEEE Visualization Conference*, pages 75–81, 1997.
[5] 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.
[6] R. A. Crawfis and N. Max. Texture splats for 3D scalar and vector field visualization. In *Proceedings of IEEE Visualization Conference*, pages 261–267, 1993.
[7] U. Dallmann. Topological structures of three-dimensional flow separations. Technical Report 221-82 A 07, Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt, 1983.
[8] 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.
[9] 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.
[10] M. Edmunds, T. McLoughlin, R. S. Laramee, G. Chen, E. Zhang, and N. Max. Advanced, automatic stream surface seeding and filtering. In *Proceedings of Theory and Practice of Computer Graphics*, pages 53–60, 2012.
[11] 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.
[12] C. Garth, X. Tricoche, T. Salzbrunn, T. Bobach, and G. Scheuermann. Surface techniques for vortex visualization. In *Proceedings of Eurographics/IEEE VGTC Symposium on Visualization*, pages 155–164, 2004.
[13] H. Guo, N. Mao, and X. Yuan. WYSIWYG (what you see is what you get) volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2106–2114, 2011.
[14] J. P. M. Hultquist. Constructing stream surfaces in steady 3D vector fields. In *Proceedings of IEEE Visualization Conference*, pages 171–178, 1992.
[15] 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.
[16] T. Klein, F. Guéniat, L. Pastur, F. Vernier, and T. Isenberg. A design study of direct-touch interaction for exploratory 3D scientific visualization. *Computer Graphics Forum*, 31(3):1225–1234, 2012.
[17] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
[18] G. M. Machado, F. Sadlo, and T. Ertl. Image-based streamsurfaces. In *Proceedings of SIBGRAPI Conference on Graphics, Patterns and Images*, pages 343–350, 2014.
[19] S. Marchesin, C.-K. Chen, C. Ho, and K.-L. Ma. View-dependent streamlines for 3D vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1578–1586, 2010.
[20] J. Martinez Esturo, M. Schulze, C. Rössl, and H. Theisel. Global selection of stream surfaces. *Computer Graphics Forum*, 32(2):113–122, 2013.
[21] T. McLoughlin, M. Edmunds, C. Tong, R. S. Laramee, I. Masters, G. Chen, N. Max, H. Yeh, and E. Zhang. Visualization of input parameters for stream and pathline seeding. *International Journal of Advanced Computer Science & Applications*, 6(4):124–135, 2015.
[22] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen. Over two decades of integration-based, geometric flow visualization. *Computer Graphics Forum*, 29(6):1807–1829, 2010.
[23] 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.
[24] F. Sadlo, R. Peikert, and E. Parkinson. Vorticity based flow analysis and visualization for Pelton turbine design optimization. In *Proceedings of IEEE Visualization Conference*, pages 179–186, 2004.
[25] 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.
[26] 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.
[27] D. Schneider, A. Wiebel, and G. Scheuermann. Smooth stream surfaces of fourth order precision. *Computer Graphics Forum*, 28(3):871–878, 2009.
[28] D. Schroeder, D. Coffey, and D. F. Keefe. 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*, pages 49–56, 2010.
[29] D. Schroeder and D. F. Keefe. 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, 2016.
[30] 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.
[31] J. Tao and C. Wang. Peeling the flow: A sketch-based interface to generate stream surfaces. In *Proceedings of ACM SIGGRAPH Asia Symposium on Visualization*, pages 14:1–14:8, 2016.
[32] F.-Y. Tzeng, E. B. Lum, and K.-L. Ma. A novel interface for higher-dimensional classification of volume data. In *Proceedings of IEEE Visualization Conference*, pages 505–512, 2003.
[33] J. J. van Wijk. Implicit stream surfaces. In *Proceedings of IEEE Visualization Conference*, pages 245–252, 1993.
[34] W. von Funck, T. Weinkauf, H. Theisel, and H.-P. Seidel. Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1396–1403, 2008.
[35] J. Wei, C. Wang, H. Yu, and K.-L. Ma. A sketch-based interface for classifying and visualizing vector fields. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 129–136, 2010.
[36] D. Weiskopf, T. Schafhitzel, and T. Ertl. Real-time advection and volumetric illumination for the visualization of 3D unsteady flow. In *Proceedings of Eurographics/IEEE-VGTC Symposium on Visualization*, pages 13–20, 2005.
[37] L. Xu, T.-Y. Lee, and H.-W. Shen. An information-theoretic framework for flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1216–1224, 2010.
[38] X. Ye, D. Kao, and A. Pang. Strategy for seeding 3D streamlines. In *Proceedings of IEEE Visualization Conference*, pages 471–478, 2005.

Jun Tao is currently a postdoctoral researcher at University of Notre Dame. 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. He is also interested in graph-based visualization, image collection visualization, and software visualization. He received the Dean's Award for Outstanding Scholarship (2015) and the Finishing Fellowship (2015) from Michigan Technological University, and a best paper award at IS&T/SPIE VDA (2013).



Chaoli Wang is an associate professor of computer science and engineering at University of Notre Dame. He received a Ph.D. degree in computer and information science from The Ohio State University in 2006. Prior to joining Notre Dame, he was a postdoctoral researcher at University of California, Davis and an assistant professor of computer science at Michigan Technological University. Dr. Wang's main research interest is scientific visualization, in particular on the topics of time-varying multivariate data visualization, flow visualization, and information-theoretic algorithms and graph-based techniques for big data analytics. He is a recipient of the NSF CAREER Award (2014), best paper awards at IS&T/SPIE VDA (2013, 2015), and an honorable mention at IEEE PacificVis (2013).