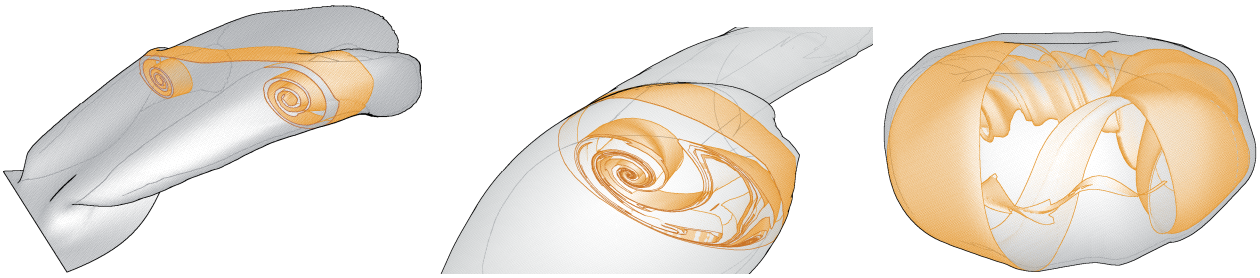
Illustrative Stream Surfaces

Silvia Born, Alexander Wiebel, *Member, IEEE*, Jan Friedrich, Gerik Scheuermann, *Member, IEEE CS*, and Dirk Bartz, *Member, IEEE CS*



(a) (b) (c)

Fig. 1. Illustrative stream surfaces in flow above a delta wing (a), a vortex breakdown bubble (b) in the same data, and a vector field with a closed orbit (c).

Abstract—Stream surfaces are an intuitive approach to represent 3D vector fields. In many cases, however, they are challenging objects to visualize and to understand, due to a high degree of self-occlusion. Despite the need for adequate rendering methods, little work has been done so far in this important research area. In this paper, we present an illustrative rendering strategy for stream surfaces. In our approach, we apply various rendering techniques, which are inspired by the traditional flow illustrations drawn by Dallmann and Abraham & Shaw in the early 1980s. Among these techniques are contour lines and halftoning to show the overall surface shape. Flow direction as well as singularities on the stream surface are depicted by illustrative surface streamlines. To go beyond reproducing static text book images, we provide several interaction features, such as movable cuts and slabs allowing an interactive exploration of the flow and insights into subjacent structures, e.g., the inner windings of vortex breakdown bubbles. These methods take only the parameterized stream surface as input, require no further preprocessing, and can be freely combined by the user. We explain the design, GPU-implementation, and combination of the different illustrative rendering and interaction methods and demonstrate the potential of our approach by applying it to stream surfaces from various flow simulations.

Index Terms—Flow visualization, Stream surfaces, Illustrative rendering, Silhouettes, GPU technique, 3D vector field data.

1 Introduction

Flow visualization plays an important role in the design process of numerous objects in science and engineering. Turbines, air planes, cars, motors, and buildings are only few examples. Although they are very different, their durability and usability is influenced by the flow behavior through or around them. The visualization of 3D vector fields can help to uncover the flow features affecting these objects. These are flow attachments, separation areas, and vortices.

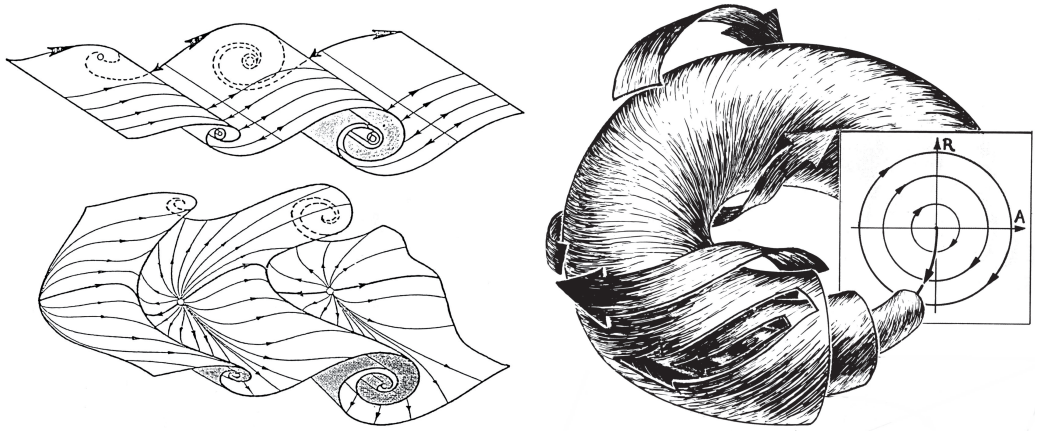
The simplest visualization in these cases are streamlines or particles. However, both lack good cues for depth perception. A widely used alternative that does not have this disadvantage are stream surfaces. An ideal stream surface is a two-dimensional continuum of streamlines starting from a well-defined space curve. It represents the area through which virtual particles pass that were released from the seed curve into the steady flow. Many variants of algorithms that approximate stream surfaces can be found in the flow visualization literature (see Section 2). Despite the fact that naïve rendering of the surface as a triangulated surface causes much occlusion, only a small fraction of available literature is concerned with improving the visualization of the surfaces.

In this paper, we will present an approach for illustrative rendering of stream surfaces. The field of illustrative rendering mimics techniques used in traditional illustrations with methods of modern computer graphics. The goal is to combine the best of both worlds. Illustrations have an age-long tradition in the communication of complex knowledge of many different scientific fields, among them flow research. Here, one of the oldest examples are Leonardo da Vinci's sketches of water flow dating back to around 1500. More recent illustrations can be found in two textbooks published in the early 1980s - Dallmann's thesis [5] and *Dynamics* by Abraham and Shaw [1]. Constrained by the lack of appropriate computer graphics algorithms at that time, they managed to break down highly-complex facts into hand-drawn illustrations of, e.g., vector fields and stream surfaces (see Fig. 2). Their images are good examples for the main accomplishment of illustrators in general: simplification of complex contexts, concentration on relevant features, and neglect of details that obstruct understanding. Due to the applied abstraction, the resulting images are both intuitive and esthetically pleasing. Up to now, many flow specialists still have Dallmann's or Abraham & Shaw's images in mind when discussing flow issues.

The main contribution of our work is a novel way of visualizing stream surfaces with illustrative methods at interactive frame rates, such that their overall shape is well-communicated and the flow char-

- *Silvia Born is with Universität Leipzig, E-mail: silvia.born@medizin.uni-leipzig.de.*
- *Alexander Wiebel is with Max Planck Institute for Human Cognitive and Brain Sciences, E-mail: wiebel@cbs.mpg.de.*
- *Jan Friedrich is with Universität Leipzig, E-mail: jan.friedrich@medizin.uni-leipzig.de.*
- *Gerik Scheuermann is with Universität Leipzig, E-mail: scheuermann@informatik.uni-leipzig.de.*
- *Dirk Bartz is with Universität Leipzig.*

Manuscript received 31 March 2010; accepted 1 August 2010; posted online 24 October 2010; mailed on 16 October 2010.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.



(a) (b)

Fig. 2. Examples for hand-drawn illustrations.: Dallmann's illustration of the deformation of a two-dimensional vortical motion (a,top) into a three-dimensional flow (a,bottom) [5] and a section of Abraham/Shaw's phase portrait of an undamped pendulum (b) are shown [1].

acteristics on the surface can be clearly depicted. We found the inspiration for our stream surface visualizations in the already mentioned works by Dallmann and Abraham & Shaw. We apply techniques that achieve similar appearance, such as contour lines, halftoning, and different cutting elements. With the combination of these methods, we are capable of producing more intuitively understandable stream surface visualizations. By employing modern computer graphics possibilities, we can go further than merely generating static text book images by achieving interactive visualizations of arbitrary stream surfaces. Provided interactions are movable cuts and slabs defined either according to the surface geometry or its flow parameters (along timelines, streamlines).

The remaining chapters have the following structure: An overview of related work concerning stream surface construction and rendering as well as illustrative flow visualization and illustrative rendering of complex objects in general is given in Section 2. The design decisions and the applied methods are presented in Section 3. After that, we present examples of our illustrative stream surfaces (see Section 4) from various flow simulations and discuss and conclude our paper in Sections 5 and 6.

2 Related Work

In this section, we give an overview of previous publications related to the two focal points of our work: stream surfaces and illustrative rendering. We dedicate a separate subsection to each of these parts because little work combining these two aspects has been reported so far.

2.1 Stream Surfaces

As mentioned in the introduction, stream surfaces consist of a continuum of streamlines. However, we will not give an overview of the large number of approaches to compute streamlines and other integral lines here. Instead, we refer the interested reader to a state-of-the-art report by McLoughlin et al. [29].

Stream Surface Construction. Stream surface construction was introduced to the visualization literature by Hultquist [17]. He presented an algorithm that approximates stream surfaces with a triangulation of a certain number of streamlines. This approximation can be refined or coarsened at converging or diverging streamlines by adaptively removing or inserting streamlines respectively. Since the introduction of the algorithm twenty years ago, a manifold body of research on the subject has achieved substantial improvements concerning adaptivity [12, 13], accuracy [12, 38, 40], topological correctness [31, 33, 39, 42], and performance [37].

Stream Surface Rendering. Despite this large body of research and although stream surfaces tend to produce visual clutter through self-occlusion, the rendering of stream surfaces has received relatively little attention so far. One of the first attempts to improve the expressiveness of stream surfaces were the stream arrows presented by Löffelmann et al. [27, 28]. In these papers, arrows cut out of the surface emphasize the direction and behavior of the flow represented by the surface and allow a view through parts of it. Laramee et al. [25] presented a combination of stream surfaces and texture-based flow visualization, where the surfaces were rendered with flow-depicting texture. Only most recent papers on the topic by Garth et al. [12] and Krishnan et al. [23] address the occlusion problems and perception issues by transparency combined with timelines on the surfaces. An earlier paper by Theisel et al. [45] also dealt with occlusion by reducing the displayed geometry to the intersection of certain meaningful surfaces. However, their approach can not improve the rendering of a single surface.

2.2 Illustrative Rendering

Illustrative visualization techniques have the potential to support scientists in processing and finding new insights into their data. Striking examples for this can be found in various research fields, such as biochemistry [51], geology [32], meteorology [19], and medicine [35, 41, 46]. Literature concerning illustrative flow and especially stream surface visualization, however, is still rather sparse and will be summarized in this section. Further, we present work dealing with the occlusion problem and visualization of complex objects in general and relate it to our approach. For surveys over standard non-photorealistic and illustrative rendering methods, we refer to Strothotte and Schlechtweg [43], Gooch and Gooch [14], and Sayeed and Howard [36].

Illustrative Flow Visualization. The already mentioned work on stream arrows by Löffelmann et al. [27, 28] is the only one addressing illustration-inspired rendering of flow surfaces. Here, the flow direction can be clearly depicted on the outer surface. Subjacent structures can be seen through the holes resulting from the cut-out stream arrows. However, the understanding of the flow on inner surface parts is still limited. We tackle this problem with interactive cutting elements for an improved view on formerly occluded structures and the flow they represent. Further publications deal with 2D flow visualization, such as Turk and Banks [48] who reproduce streamline placement as seen in hand-drawn illustrations or Kirby et al. [22] who adapted concepts from oil paintings. Haloed line renderings - known from traditional line illustrations - and its application to streamlines (in 3D) have been presented by Everts et al. [10]. The illustration-inspired visualization of time-varying volume data has been addressed as well. Svakhine et

al. [44] emphasize structural information in the flow volume by applying silhouettes and methods from photographic flow visualizations like Schlieren. Joshi et al. [20] on the other hand give hints on flow directions by using cartoon techniques, as, e.g., speed lines and strobe silhouettes. Hsu et al. [16] visualize flow evolution over time by introducing illustrative methods, such as silhouettes and temporal fading.

Illustrative Rendering of Complex Objects. As already stated above, in stream surface rendering one main challenge is self-occlusion and the visualization of several layers. The same is the case in the field of technical illustration, since its motifs (such as, e.g., motor blocks, cars) are equally complex. Here, line renderings (depicting several layers) with some degree of surface shading are the most widely used technique, which we apply as well. Fundamental work concerning line renderings was presented by Dooley et al. [8] who set down illustration rules for line renderings: line width, transparency, and style (e.g., dashing) depict different line characteristics, such as importance, layering, or silhouette type. Appel [2] discussed haloed line visualizations to improve depth perception in line renderings. In our method, we distinguish layers by attenuating the line color with increasing depth.

Examples of line renderings in technical illustrative visualizations include the work by Nienhaus and Döllner [30], Kaplan et al. [21], and Fischer et al. [11], who use several silhouette layers to convey the inner construction of technical models. Surface shading is either omitted [21] or unobtrusive methods like transparent rendering [7, 30] or halftoning [11] are applied. We use halftoning, which is recommended by Dooley et al. [9]. They discuss techniques to efficiently render nested surfaces and prefer hatching-like surfaces to transparent renderings [7, 30] because of the improved depth representation of the layering.

Apart from cleverly combining line and surface renderings, occluded structures can be displayed with various illustrative compositing techniques. For a complete overview, we refer to Viola and Gröller [49]. These smart visibility methods differ mainly in the way they represent occluding structures, i.e. either by deforming the model (peel-away, exploded view) or by removing them completely (cut-away view) or to some extent (ghosted view). Only variants of cut-away and ghosted views are applicable to stream surfaces. Exploded views are designed for the display of complex models consisting of various units [3]. Since stream surfaces usually consist of a single part, exploded views are inapplicable for their visualization. Applying deformations, as used in peel-away views [4], to stream surfaces would also modify the flow information, which may lead to misinterpretations. Cut-aways and ghosted views [26], however, reveal subjacent layers and still provide sufficient context information. We modified the cut-away and ghosted views concept by cutting out sections of the surface and visualizing these in the context of the whole surface. By using cuts defined by geometrical properties, inner shape is made visible. Cutting according to flow characteristics allows to depict and track flow.

3 Illustrative Stream Surfaces

3.1 Visualization Design

Requirements. When aiming at an intuitive stream surface visualization, it is essential to identify and capture stream surface characteristics necessary for the viewer to understand the specific flow situation: First, since a stream surface is an approximation of a seed line's evolution over time, the *overall shape* of the stream surface is a main clue for understanding the flow. However, stream surfaces are usually non-trivial surfaces with a high degree of self-occlusion. Therefore, rendering and interaction techniques must be provided for the enhancement of shape and depth perception, especially in the case of usually hidden parts of the surface, such as, e.g., inner windings or vortex breakdown bubbles. Second, *flow features on the stream surface* are of importance, because shape by itself cannot communicate the flow situation completely. Flow direction, courses of time- and streamlines as well as the location and characteristics of singularities, such as sources and sinks, are of interest. Here, methods must be taken into account which can depict flow directions on all parts of the surfaces without leading to visual clutter and thereby impairing the overall shape perception.

How these are met in traditional illustrations. Both, Dallmann's thesis as well as Abraham & Shaw's textbook, have the status of a standard work in the field of flow research. Their images set the benchmark for our visualizations and their applied illustration methods serve as the starting point for our rendering techniques. In Figure 2a, examples of Dallmann's illustrations show the *overall shape* by silhouettes and feature lines. Though hidden lines are also drawn at specific positions. The dashed appearance indicates that they are not visible from the current point of view. Different textures or colors are used to help differentiate between the front and backside of the stream surface. *Flow direction* is mainly represented by streamlines with arrow heads drawn onto the surface and contour lines. These are especially useful when complex flow features appear, such as the sources in Figure 2a (bottom). Note that streamlines not only give information about the flow directions, but also about the surface curvature and thereby improve shape perception. The same is the case with the hatching technique applied in Abraham & Shaw's illustration (see Fig. 2b): strokes not only depict the flow direction, but also the torus shape. Cutting away different parts of the stream surface is a method often applied to allow insight into inner surface parts (see Section 2.2). In Figure 2b, the different layers of the torus are represented by successively peeling away the outer layers. Further, *Poincaré sections*[1] or similar cuts are used by Abraham & Shaw to show the inner layers and the according flow direction.

Design decisions based on requirements. Our goal is to enhance the visualization possibilities of stream surfaces and support flow researchers in their investigations by meeting the requirements discussed above. Since stream surfaces are very diverse concerning shape, size, and complexity, we do not propose a single visualization technique applicable to all stream surfaces in general. Instead, we aim at providing the flow researcher with a tool consisting of a suite of different illustrative rendering and interaction techniques. With that, the user can explore and interact with stream surfaces without further preprocessing and adapt the visualization depending on the currently observed surface and the flow aspects he is interested in. Thus, in contrast to the illustrations presented above, we do not only provide static images, but exploit the advantages of computer graphics and allow real-time interaction in 3D. We adapted all techniques to be GPU-based and thereby allow for a real-time switch between different rendering techniques and a comfortable interaction without delays - even for large stream surfaces.

In accordance with the demands discussed above, we provide visualization techniques representing the *overall shape* and *flow features*: The basic shape is provided by silhouettes, features lines, and halftoning without introducing too much visual clutter and occluding inner structures. Flow information on the surface is depicted by illustrative surface streamlines. The problem of visualizing shape and flow characteristics on inner and usually occluded parts of the stream surface is solved by introducing movable cuts and slabs. By moving them interactively, the user can explore the surface shape and the flow it represents. In the following, the implemented methods are described and discussed in more detail:

- *Silhouettes and features lines* depict the main shape of the stream surface. In addition to the shape information of the front layer, the silhouettes of the second and rear surface layer give shape information of otherwise occluded surfaces. With increasing

[1] **Poincaré sections** are special cuts used for illustration and analysis of periodic orbits in dynamical systems. They show the repeated intersections (Poincaré map) of the periodic orbit with a lower dimensional subspace (e.g. a plane) and thus enable the depiction of the orbit's behavior over time [15]. Different patterns of the intersections allow to distinguish different types of periodic orbits: attracting, repelling, saddle-like and spiraling behavior are common.

depth, the silhouette colors become less saturated thus more similar to the white background color. This depth cue is called *proximity luminance covariance* [50], mimics atmospheric depth, and hence allows a distinction between the different silhouette layers. In contrast to rendering nested surfaces with varying transparencies, this method gives unobtrusive context information, which can be easily combined with the techniques described below. The display of the individual layers can be controlled by the user.

- *Halftoning* is a non-photorealistic shading technique, that we apply as a discreet way of giving further shape and depth information, especially in more complex surfaces. Different colors can be used for the back and front faces of the surface, which allows an easier understanding of, e.g., narrow windings. Its advantage is that it minimizes distraction because colors of subjacent surfaces remain unchanged and the depth ordering of nested structures remains clear. Both can be problematic with the alternative of transparent surface renderings [9].

- *Illustrative surface streamlines* are ribbon-like structures with arrow heads depicting flow direction and surface shape (similar to the streamlines in Fig. 2a). Their appearance (e.g. width) and the number of arrows per streamline can be adapted by the user. The illustrative surface streamlines are constructed with a GPU-based method using a geometry shader. For this, a parameterization of the stream surface is necessary. In contrast to dense texture-based visualization techniques [25], this method allows the depiction of flow direction on the surface without occluding inner structures.

- *Cuts* are represented by interactively movable lines, which are useful for the exploration of the surface shape and the flow it represents. The cuts are either defined based on geometry or on the flow parameters s and t given for each surface vertex. Geometry-based cuts correspond to the intersection line of a user-defined cutting plane and the stream surface. This provides insight into the inner shape of the surface and Poincaré sections can be visualized for periodic orbits (see Fig. 13). Flow parameter-based cuts on the other hand represent single streamlines (defined by an s-isovalue) and timelines (defined by a t-isovalue). By moving these parameter-based cuts, either timelines on the surface can be tracked over time or streamlines can be explored for different starting points. The flow direction on the cuts can be depicted by additional arrow heads.

- *Slabs* are an extension of the cuts in the sense that - instead of lines - stream surface sections of user-defined width are displayed. Analogously to the cuts, a geometry-based slab results in a straight section through the stream surface (see Fig. 1a,b), whereas a flow parameter-based definition produces a strip of streamlines or timelines respectively. By moving the parameter-based slabs, the user can track the flow over the stream surface (either by following timeline strips over time or by observing streamline strips for different starting points), which is not possible with existing methods so far. The rendering techniques described above (silhouettes, feature lines, halftoning, illustrative surface streamlines) can be applied to these slabs as well, which enhances shape perception in comparison to cuts.

3.2 Method Description and Implementation

To provide the proposed real-time interactivity, the previously presented visualization design is implemented on the GPU. An overview of our algorithm is given in Figure 3. It consists of several preparation steps (indicated by the gray boxes) and one final composition step (blue boxes). The user can combine the desired visualization features during runtime. According to these user specifications, the preparation steps are executed. The intermediate results, which are, e.g., silhouettes, surface, or slab style, are rendered into textures and made available to the shader performing the final composition in image space. The blue boxes in Figure 3 depict this shader's workflow and show that the different features are assembled from back to front.

Some techniques are based on the geometrical features of the stream surface whereas others need flow parameter information (e.g., illustrative surface streamlines or the flow variants of the cuts and slabs). These parameters are delivered to the shaders as the stream surface's texture coordinates. In the following, the different visualization features and their implementations are explained in more detail.

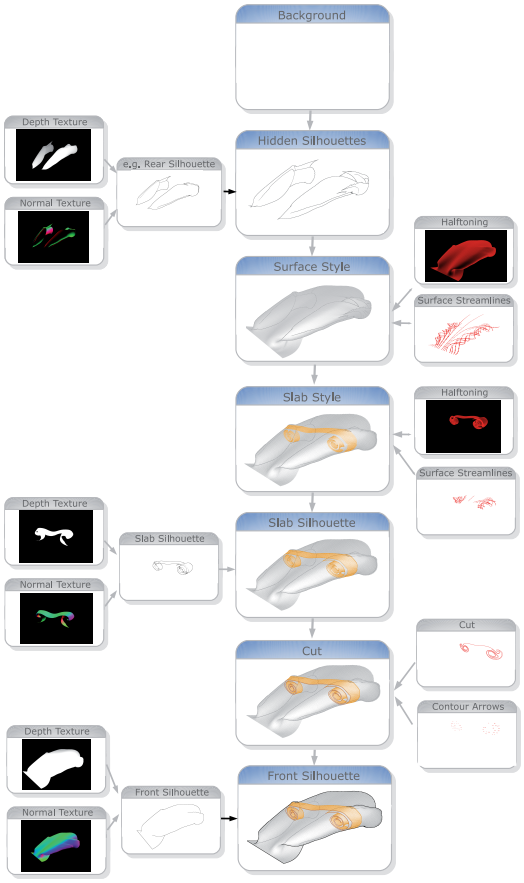


Fig. 3. Overview over the required rendering steps. The gray boxes resemble the intermediate results of the preparation steps for the different illustrative features. The textures are delivered to the final composition stage (blue boxes). This shader assembles the scene from back to front.

3.2.1 Silhouettes

Silhouettes are the main cue for shape recognition of 3D objects and are well-suited to give context information when used with other methods, as, e.g., shading techniques. According to Gooch and Gooch [14], silhouettes are defined as points on a (smooth) surface with normals perpendicular to the viewing direction. With polygonal models, however, silhouettes can be approximated as edges being part of a front- and a back-facing polygon. With this definition, the contour lines separating the model from the background as well as internal silhouettes can be described. Further feature lines necessary for shape recognition are creases (discontinuities on the surface, such as the edges of a cube) and border lines (edges belonging to only one polygon). They occur in non-closed meshes and thus also in stream surfaces.

Silhouette Detection. Isenberg et al. [18] gives an overview over silhouette detection algorithms and divides them into *image-space*, *object-space*, and *hybrid* algorithms. They differ in performance, precision, and the resulting appearance of the silhouette. Since we aim to achieve interactive frame rates, but do not require stylized silhouettes or subpixel precision, we base our silhouette detection on the

hybrid algorithm of Decaudin [6]. This algorithm has become a standard method, because it detects all silhouettes and feature lines and solves the hidden line problem implicitly. Here, silhouettes are determined based on depth changes (mainly for outer contour) and normal discontinuities (mainly for internal silhouettes) in the scene. We implemented the algorithm with two rendering passes. In the first pass, the stream surface's depth values are stored in a depth texture whereas the transformed normals are stored in a normal texture. The latter is an RGB-texture containing a pixel's normal vector as its color value. In the second pass, silhouettes are detected in these textures: A Sobel edge detection filter is applied to the depth texture calculating the depth gradient for each pixel. If this gradient exceeds a certain threshold, the pixel is classified as silhouette. In the normal texture, silhouettes are detected by checking for normal discontinuities, i.e. normals of neighboring pixels are not parallel. Thus, for every pixel the dot product of its normal with the normals of its four direct neighbors is computed. A dot product smaller than one indicates that two normals are not parallel. Thus, if the sum of these four dot products drops below a certain threshold (we used 3.95), a discontinuity in the surface and by that a silhouette is detected. The final silhouette result is stored in a texture, which is initialized with white and set to black for all pixels classified as silhouette either in the depth or in the normal texture (*Rear Silhouette* and *Front Silhouette* in Fig. 3).

Layer Management. The general silhouette detection presented in the last paragraph can be used to capture silhouettes from different surface layers (front, second, and rear; see Fig. 4). For that, the normals and depth values of the designated layer have to be rendered into a texture in the first pass. This is controlled by appropriate depth tests. For the *front* layer, the standard z-buffer test is used. The *rear* surface is captured by rendering only fragments with the highest depth values by changing the standard z-buffer depth test from GL_LESS to GL_GREATER. For the *second* surface layer *depth peeling* [30, 11] is applied. For that, each fragment of the stream surface has to pass two depth tests. It has to be at a greater depth than the front layer. This peels away all fragments of the front layer. After that, the remaining fragments have to pass the z-buffer test. Together, this results in the rendering (and silhouette detection) of the second layer.

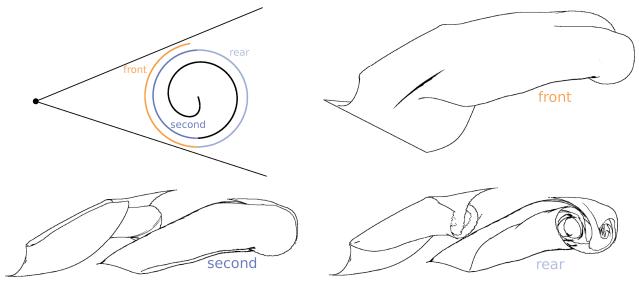


Fig. 4. The different silhouette layers. Upper left shows the scene layout and the computed silhouette layers: front (orange), second (dark blue), and rear layer (light blue). The other three images show the intermediate results of front (upper right), second (lower left), and rear surface layer (lower right) of a stream surface of the delta wing dataset.

3.2.2 Halftoning

Halftoning traditionally denotes the preparation of newspaper images for black-and-white printing. Continuous tone images are reproduced by filling small resolution units with black dots of varying size. For modern graphical output devices, these resolution units are composed of pixel areas of varying size ($n \times n$). Different tones are achieved by controlling the number of pixels set in this unit according to a specific pattern (*ordered dithering*). Similarly, halftoning can be used for surface shading. Here, the pixels of predefined areas are set depending on the current lighting. One advantage of this shading technique is that only a portion of the surface pixels are set, such that subjacent structures are still visible and their colors are not modified.

We apply the *procedural screening* technique described by Strothotte and Schlechtweg [43]. In contrast to, e.g., ordered dithering, there is no need to provide the pixel pattern (called *dither screen*) as input texture. Instead, the dither screen is defined by a dither kernel $\tau(u,v)$ and a mapping function $M(x,y)$. The dither kernel controls the appearance of the halftoning texture, whereas the mapping function defines its scale and orientation in image space. The technique is implemented as follows. First, a pixel's image space coordinates (x,y) are transferred to dither texture coordinates (u,v) by applying the mapping function M.

$$(u,v) = M(x,y) = \left(\frac{x' \bmod n}{n}, \frac{y' \bmod n}{n}\right) \quad \text{with} \quad (x',y') = R_\alpha \cdot (x,y) \tag{1}$$

where R_α is a rotation matrix, that we use to achieve a slightly tilted appearance of the halftoning pattern. To obtain valid dither texture coordinates (u,v), the result is clamped to $[0,1]$ by taking into account the user-defined texture side length n (values from two to four lead to good effects). The dither kernel $\tau(u,v)$ assigns a halftoning threshold to each dither texture coordinate (see Equation (2)). I_{cross} is the intensity threshold for the switch from parallel lines to crossed lines. Finally, for each pixel the current lighting intensity is compared with the threshold at the corresponding dither texture position. The pixel is set, if the lighting falls below the dither threshold (see Equation (3)).

$$\tau(u,v) = \begin{cases} I_{cross} \cdot v, & u \le I_{cross} \\ (1 - I_{cross})u + I_{cross}, & \text{else} \end{cases} \tag{2}$$

$$pixel(x,y) = \begin{cases} 1, & I_{pixel}(x,y) < \tau(u,v) \\ 0, & \text{else} \end{cases} \tag{3}$$

For the lighting, we use the non-linear shading model of Krüger et al. [24]:

$$I = a_l a_m + \left(\frac{L \cdot N + 1}{2}\right)^\alpha \cdot d_l d_m \tag{4}$$

L is the lighting direction at one surface point, N is the surface normal. The terms a_l, a_m and d_l, d_m describe the ambient and diffuse lighting and material intensities respectively. Similar to a gamma correction, the accentuation of different tones can be controlled with α. For $\alpha < 1$, the middle tones are emphasized and the light and dark tones are compressed, leading to a rather uniform result. For $\alpha > 1$, more contrast is achieved since light and dark tones are accentuated. With this shading model, the appearance of the halftoning can be adjusted without changing the dither kernel characteristics. The result of this rendering step is stored in a texture (*Halftoning* in Fig. 3) and provided to the final composition shader.

3.2.3 Illustrative Surface Streamlines

Illustrative surface streamlines are ribbon-like streamlines with arrow heads, which indicate flow direction and singularities on a stream surface (see Fig. 2a). A stream surface is a continuum of streamlines starting from a seed curve and extending over time. Information about the represented flow is given by a parameterization of the surface (additional scalar values for each vertex). The parameter t depicts time and increases along the streamlines. The parameter s increases along the seed curve (see Fig. 5). We utilize a geometry shader to detect the isolines on the stream surface and create the ribbon-like geometries along these lines. A geometry shader is executed in the rendering pipeline between vertex and fragment shader and is capable of emitting new geometry derived from the input primitives. In our case, the shader receives the stream surface triangles, the corresponding flow parameters s and t for every vertex (transferred as texture coordinates), and user-specifications concerning the number and appearance (width, number of arrow heads) of the streamlines as input. Its output are triangles representing the illustrative surface streamlines evenly distributed over the stream surface. For further use in the final composition stage, the newly created geometry is rendered into a texture by the fragment shader (*Surface Streamlines* in Fig. 3).

The streamline construction itself is implemented in the geometry shader as follows (see Fig. 5, middle part). A streamline is an isoline defined by a specific s-parameter value s_{ribbon} (resulting from the user-defined number of streamlines and the s-parameter range of the surface). Each triangle (purple in Fig. 5) is tested whether its edges cross one of the streamlines to be constructed. If a crossing edge is detected, the position and t-parameter of its cutting point c with the streamline is computed. If two cutting points c and c' per triangle are detected, their connecting edge represents a section of the ribbon's centerline (green line). For every such centerline section, the respective ribbon segment is generated by forming a trapezoid such that the triangles (green) border the current triangle and are parallel to the centerline.

Two special cases have to be handled. First, the centerline might coincide with triangle edges (see Fig. 6a). In this case every centerline segment is visited twice because of its two adjacent triangles (large yellow). To avoid that the surface streamline section is generated twice, only the segment inside the current triangle is generated (blue triangles with yellow border). Gaps occur at triangles (large red) with only one cutting point. For these, appropriate triangles have to be emitted by the shader (blue triangle with red border). The second special case is the handling of diverging streamlines. This occurs at positions where the mesh resolution changes and a triangle has three cutting points (see Fig. 6b). The cutting points must be correctly transformed into a branching (red line) or else the two neighboring streamlines are connected. This leads to a loop and an interruption of the correct streamline course (red dashed line).

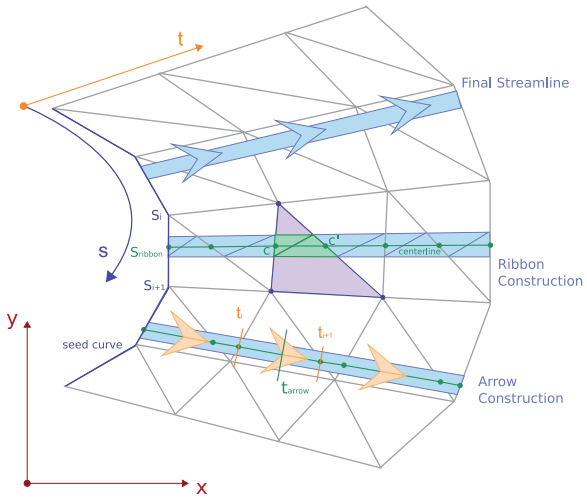


Fig. 5. Construction of illustrative surface streamlines. The light gray grid represents a parametrized stream surface (parameter t depicts time, parameter s increases along the seed curve). A complete illustrative surface streamline is shown in the top part. In the middle, the streamline construction is shown. The edges of a triangle (purple) are checked for crossings with a streamline (defined by s_{ribbon}). The cutting points (such as c,c') define the centerline (green) of the streamline and two new triangles (green) form the final streamline segment. Then, arrow heads are added (bottom). The interpolated t-values t_i, t_{i+1} of the cutting points are checked, whether an arrow head (at position t_{arrow}) needs to be added between them. If yes, the exact position of the arrow head on the streamline is determined (by interpolation) and two triangles (yellow) forming the arrow are created.

The arrow heads for the illustrative surface streamlines are generated analogous to the ribbons. The position t_{arrow} of an arrow head results from the user-defined number of arrows and the known t-parameter range. For every centerline segment, it is queried whether t_{arrow} should be between t_i and t_{i+1} of the two cutting points (see Fig. 5, bottom). If this is the case, the arrow's center point on the streamline is derived by interpolation. The arrow head itself consists of two triangles defined by the center point and the given arrow width.

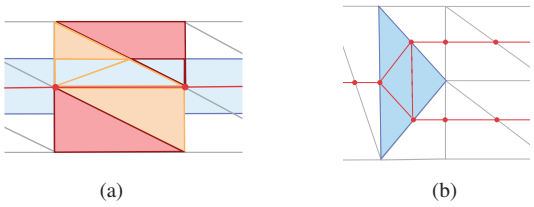


Fig. 6. Special cases. (a) The centerline might coincide with triangle edges (large yellow triangles). To avoid that the ribbon segment is generated twice, only the part in the current triangle is drawn (blue triangles, yellow border). To avoid gaps at positions with triangles (red triangle) with one cutting point, appropriate triangles are inserted here (blue triangle, red border). (b) Diverging streamlines cause changes in mesh resolution of the stream surface. The blue triangle has three cutting points, which need to be transformed into a branching instead of the connection of neighboring streamlines (dashed red line).

3.2.4 Movable Cuts and Slabs

Cuts and Poincaré sections are used in illustrations to give insight into hidden parts of a stream surface. We apply and enhance this visualization technique by making it interactive, expanding it to slabs, and providing two variants, a *geometry-based* one to explore the shape and a *parameter-based* one to observe the flow. The difference is also depicted in Figure 7.

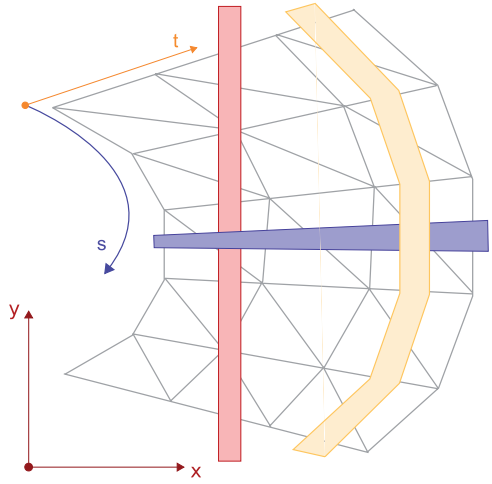


Fig. 7. A geometry-based cutting plane is defined based on the Cartesian coordinate system (depicted in red, in 2D for simplicity). The red slab is an example for a geometry-based cutting plane moving along the x-axis. A parameter-based cut is defined based on the s- or t-parameter. Thus, it is either oriented along timelines (yellow) or streamlines (blue).

Cuts and Poincaré Sections. The two-dimensional cuts are constructed with the use of a geometry shader. Its inputs are the surface triangles, the current cutting mode (i.e. parameter-based or geometry-based) and a position on the desired main axes - both specified by the user. The cut detection itself is analogous to the surface streamline detection in Subsection 3.2.3. Each triangle is tested whether its edges cross the predefined cutting plane. If two crossing edges are detected for a triangle, the cutting points are determined and connected. The shader output is a line strip resembling the cut, which is rendered into a texture (*Cut* in Fig. 3) by the fragment shader. The geometry-based and parameter-based approaches differ only in the vertex information that is used for the query (either vertex coordinates or flow parameters). Our implementation of Poincaré sections are a special case of the geometry-based cuts. Here, a periodic orbit can be investigated by moving a cut around the orbit in a circular way.

Therefore, the position of the cut is defined by an angle (geometry shader input) instead of a location along the main axes.

A further feature of traditional illustrations are arrows depicting flow direction on Poincaré sections (see Fig. 2b). We reproduced this and applied it to our cuts. The contour arrows are generated in a separate geometry shader, which distributes them evenly on the corresponding cut according to a user-defined arrow density. The direction of the arrow heads is computed based on the t-parameter gradient at a specific position (*Contour Arrows* in Fig. 3).

Slabs. A slab is an extension of the cut and represents a wider, interactive surface slice. It is rendered with the same techniques as the complete stream surface (as described in the Subsections 3.2.1-3.2.3) to enhance shape perception of the slab. The computation of the slabs is accomplished in the fragment shaders of the respective techniques by discarding all fragments positioned outside the slab (given by user-defined position and width). Thus, analogous to the complete stream surface, intermediate results for the slab style (halftoning and surface streamlines), and for the slab silhouette (all depicted in Fig. 3) are generated and used in the final composition. Again, the parameter-based and geometry-based versions differ only in the vertex information that is used for the clipping decision: either the vertex coordinates or its flow parameters.

4 Results

In this section, we apply the presented methods to a number of complex datasets and provide performance numbers for all examples.

4.1 Delta Wing

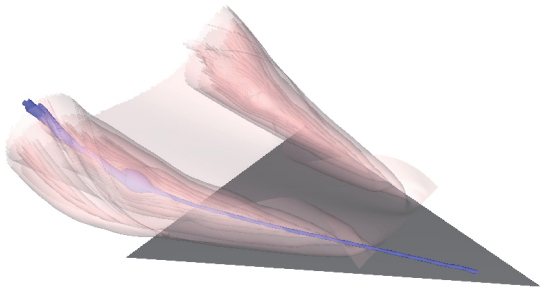


Fig. 8. An overview of the delta wing dataset with two stream surfaces.

Our first examples are two stream surfaces from a simulation of airflow around a single delta-type wing configuration. The dataset was computed in the context of numerical research of vortex breakdown. Its main features are a system of vortices and vortex breakdown bubbles with recirculations zones above the wing. The rendering of the surfaces is heavily influenced by the complex flow situation. The recirculating flow in the breakdown bubbles (Figure 9) as well as the rolling up of the vortices (Figure 8) lead to a high degree of self-occlusion of the surface.

Conventional visualization of the breakdown bubble using transparency and coloring in the t-direction is presented in the images of Figure 9a,b. The first image clearly shows that the surface is strongly folded and the bubble is built up of many layers. The second image provides information about the flow direction on the outer surface by varying the color along the streamlines. However, the flow direction is quite unclear in both images. It is very hard to distinguish the different layers and to follow the flow when inspecting the translucent surface. Figure 9d shows how our illustrative rendering combines silhouettes and halftoning (to provide context) with an s-parameter-based slab. The latter is a continuum of streamlines conveying the flow direction on several layers over time. The user can see that the flow enters on one side (lower left corner of the image) and curls its way through the bubble before leaving it again on the other side (upper right). This image does not show all layers as this would be confusing in this type of illustration. Using an interactive geometry-based slab as in Figure 9c and Figure 1b is the alternative we provide to inspect the different layers nevertheless. They are clearly discernable in both figures and reveal the inner structure of the bubble that was hidden in the previous depictions. See the comparison in Section 5 for the best alternative.

Figure 10 strikingly demonstrates that we achieved the goal to come close to hand-drawn pictures from text books: our illustrative visualization has the same streamlines with arrows, silhouettes, and halftoning. It even allows the illustration of the vortex at the back part of the stream surface. The hand-drawn images definitely emphasize the important features. The comparison ensures that our rendering is at least as expressive as the illustration by Dallmann.

Finally, Figure 1a and Figure 11 show exemplary illustrative renderings for a surface covering both vortices. Figure 11a uses the contour of the surface combined with a geometry-based cut (orange) and halftoning. It is enhanced by showing the direction of the flow by illustrative surface streamlines (gray). The main features of the vortices are more prominent in this image than in the transparent surface rendering of Figure 8. Figure 11b provides insight into the inner flow along the streamlines of a vortex. A visualization with a geometry-based slab as in Figure 1a on the other hand depicts the inner shape and windings of the vortices. All in all the combination of the presented techniques yields views into the inner structure of the vortical features of this dataset that have not been possible before.

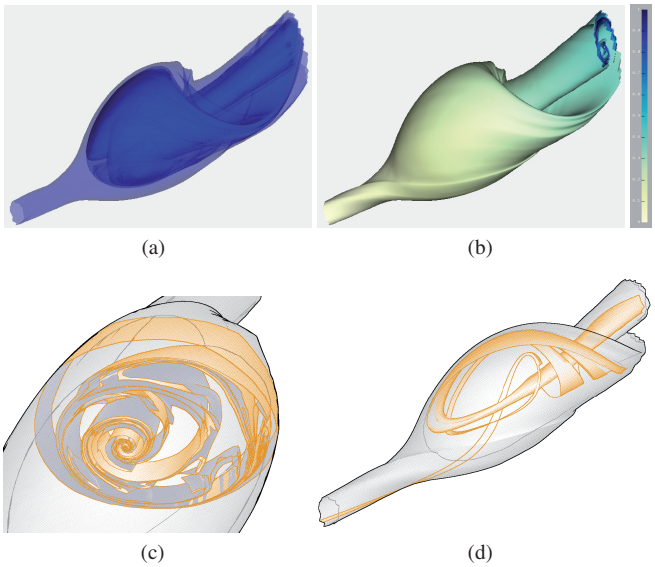
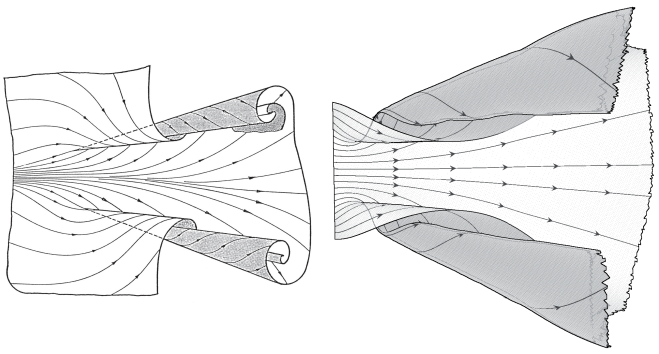


Fig. 9. Comparison of four different renderings of the same stream surface depicting a vortex breakdown bubble of the delta wing dataset: Translucent (a), color-coded t-parameter (b), illustrative with geometry-based slab (along x-axis) showing the different folded layers of the stream surface inside the bubble (c), and illustrative with parameter-based slab (d). The occasional gaps in the halftoning result from flipped normals or holes in the surface.

4.2 Closed Orbit

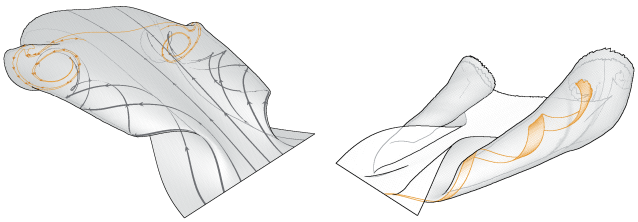
In contrast to the previous example, the closed orbits used in this section are created synthetically. However, it is well known that such orbits exist also in real world computational fluid dynamics data [34]. In our case, we can see a closed or periodic orbit as a streamline that is closed, i.e. that runs in a cycle, passing the same positions in space over and over again. In all cases presented in this section, the closed orbits act similar to sinks, i.e. streamlines from nearby are attracted. The stream surfaces are all started from circles as seed lines. A non-attracting version is illustrated by Abraham & Shaw as reproduced in Figure 2b. They show stream surfaces with stream arrows together with a flow cross section depicting the movement of the recurring streamlines, that is a Poincaré section.

Our techniques are able to illustrate the characteristics of the flow by using slabs and surface contours in Figures 1c and 12. Both im-



(a) (b)

Fig. 10. Dallmann's illustration (a) compared to our visualization (b) of a stream surface for the delta wing.



(a) (b)

Fig. 11. Delta wing: Stream surface with illustrative surface streamlines (gray) and a cut (orange) showing the rolling up of the surface (a) and a similar stream surface with a slab representing a continuum of neighboring streamlines showing the flow in the inner part of the vortex (b).

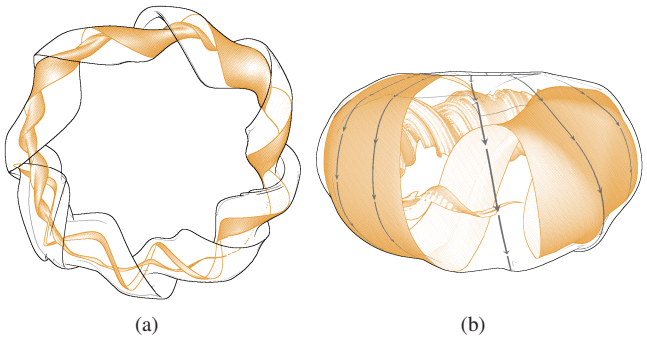
ages in Figure 12 show parameter-based slabs along streamlines. In Figure 12b further information about the flow direction is given by illustrative surface streamlines. These visualizations unfold their value best when moved interactively by the user. Nevertheless, they provide a good overview of the flow also in still images. Both nicely show the nesting, twisting, and swirl of the stream surface and thus the flow.

As we already mentioned, Poincaré sections are a valuable tool for the analysis of closed orbits. Figure 13 shows our illustrative analog applied to one of our datasets. The nesting of the surfaces as well as the attracting nature of a closed orbit are clearly visible in the flow cross section. The arrows give the needed hints to the direction of the flow along the rendered contours. In the case of very tight windings, the structure is a bit clearer without the vectors. But even then, we leave this up to the user who can switch the arrows off and on (e.g. when viewing a close-up) during their exploration of the surface. The widespread use of Poincaré sections in the fluid dynamics community makes our visualization intuitive and easy to use for domain experts.

4.3 Combustion Chamber

A simulation of the flow in a combustion chamber serves as data for the stream surfaces in this subsection. This kind of chamber is used in central heating systems for houses. While exhaust gas exits the chamber on the right end (see Fig. 14), an inlet for the gas is situated on the opposite side (blue in Fig. 14). Nine inlets for air are distributed along each of the side walls. This construction leads to a high degree of turbulence and many vortices and singularities in the vector field. The turbulence is desired to achieve the good intermixture of air and gas needed for combustion.

We computed stream surfaces from two three-dimensional spiral saddle points (see Fig. 15). The seed lines were chosen to lie in the unstable manifolds of the saddles. Thus, both stream surfaces move away from the saddle points in a spiraling manner. In the images, this behavior is illustrated by augmenting the surfaces with two of the proposed techniques: illustrative surface streamlines and slabs. Both



(a) (b)

Fig. 12. Twisted closed orbits with parameter-based slabs along streamlines. In image (b) flow direction is depicted by additional illustrative surface streamlines.

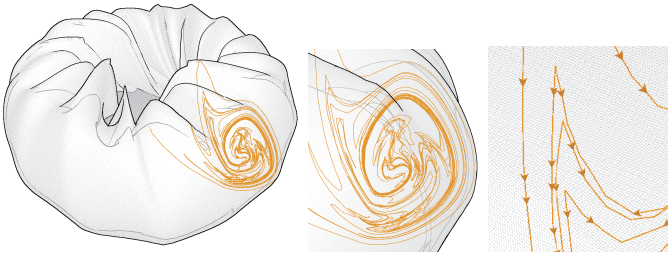


Fig. 13. Twisted closed orbit. Poincaré section-like flow lines in a slice orthogonal to the main flow direction. The close-ups show the same position in the orbit and a detail with contour arrows.

techniques nicely illustrate the divergence and rotation and thus the nature of the saddle point. The slab additionally allows the user to follow a part of the flow even into hidden layers of the surface. Especially in Figure 15b, one clearly sees how the stream surface returns to an area close to the saddle point. A t-parameter-based slab is applied in Figure 15a displaying a continuum of timelines, i.e. shows where seeded particles are at a certain point in time. Note that the slab is interactively adjustable in width and position, allowing for an intuitive exploration of a slice of timelines over the complete surface.

4.4 Performance

Interactivity is an important prerequisite for a comfortable exploration of stream surfaces. We measured performance of our illustrative rendering approach for the previously presented datasets in low ($800x600$) and high resolution ($1800x1000$) on a computer with an Intel Core 2 Quad Q9450 CPU and an Nvidia GeForce GTX 260 graphics card. The frame rates are listed in Table 1 and show that interactivity is

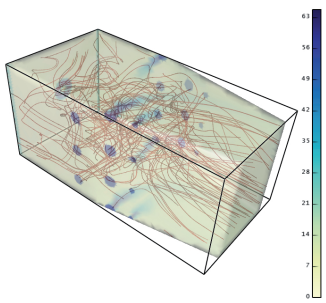
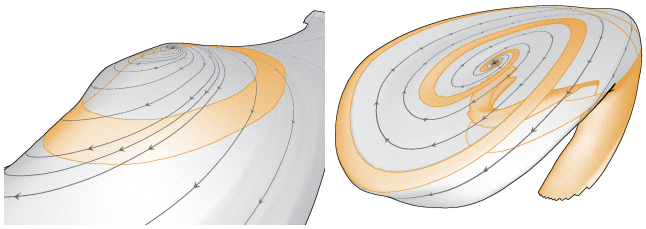


Fig. 14. An overview of the combustion chamber with streamlines and color coded velocity magnitude on the boundary (white to blue). Be aware that the colors are paler in the image than in the color bar due to the translucency.



(a) (b)

Fig. 15. Two stream surfaces starting at spiral saddles (unstable manifolds) in the combustion chamber dataset. Each image has arrows along the streamlines and a slab along the timelines (a) or the streamlines (b) respectively.

Table 1. Frame rates (in frames per second) in low ($800x600$) and high resolution ($1800x1000$) for the datasets presented in Sections 4.1 to 4.3.

Dataset (Figure)	#Vertices	FPS (low res.)	FPS (high res.)
Delta wing			
1a	33k	168	89
10b	37k	116	71
11a	37k	125	70
11b	37k	153	85
Breakdown Bubble			
1b / 9c,d	133k	88	61
Closed Orbit			
1c	396k	40	22
12a	388k	42	26
12b	396k	37	20
13	396k	58	44
Combustion Chamber			
15a	124k	69	36
15b	47k	85	57

achieved for all examples. Naturally the performance decreases with higher resolution and larger models (such as the closed orbits), but still allows comfortable exploration.

5 Discussion

Comparison with Existing Methods. Although the results section already compared our methods to previous approaches, we would like to emphasize two points. First, the only techniques providing a similar depth of insight into the structure of vortex breakdown bubbles is the work by Tricoche et al. [47]. They, however, used a combination of direct volume rendering and vector field topology in moving section planes, i.e. no stream surfaces, and their approach is not interactive. Second, the earlier mentioned approaches by Garth et al. [12] and Krishnan et al. [23] yield expressive visualizations, but their approaches are not interactive in the way that users can modify the rendering in the way most suitable for their application.

Evaluation. We presented our results to a fluid mechanics expert whose special subject is vortex breakdown. He regards all the visualizations suitable. He specifically considers the vortex breakdown bubbles to be a significant improvement over existing visualizations. With the inner layers of the bubble, the visualization demonstrates the proportions of rolled-up structures, rapid changes, and its curvature and twist.

Limitations. As our method uses a large variety of different visual cues for the communication of the surface structure, we believe that additionally coloring the surface with other quantities would make the visualization overly complex. However, because our tool is completely interactive we can simply switch between the presented and standard rendering techniques. We therefore recommend the use of illustrative stream surfaces for the exploration of the surface structure and a switch to standard rendering with color coding for inspecting additional quantities at certain locations of the surface.

6 Conclusion

The presented combination of stream surfaces and illustrative rendering techniques proved to be a very expressive tool for the visualization of complex flow structures. It closely resembles hand-drawn flow images from renowned text-books whose value is undisputed. The application of the methods to real world datasets showed the usefulness and applicability for the analysis of data produced by modern computational fluid dynamics simulations. Silhouettes and illustrative surface streamlines provide overview of the surfaces with their flow directions, slabs give a good orientation for surfaces with multiple layers, and Poincaré section-like slices reveal the complete folded structure of complex flow patterns. Furthermore, all techniques are completely interactive and thus support explorative inspection by domain scientists. We believe that the resulting images have the same inherent beauty as their hand-drawn counterparts.

As, in principle, the presented methods can be applied to any kind of surface parameterized by two variables, we will explore the usefulness for other surface types in the future. The focus will lie on streak surfaces and path surfaces, as we know that their characteristics fit the prerequisites of our methods.

Acknowledgments

We are grateful for the help of Albert Pritzkau while preparing the supplemental material. We would also like to thank Markus Rütten of the German Aerospace Center (DLR) in Göttingen for providing the simulation datasets and for many helpful comments during the evaluation of the presented techniques. We are grateful to Wieland Reich for providing the synthetic periodic orbit. Special thanks go to the *FAnToM* development group for providing their stream surface algorithm. This work was partially supported by DFG grant SCHE 663/3-8.

We dedicate this paper to our friend and colleague Dirk Bartz, who passed away suddenly and very unexpectedly on March 28^{th}, 2010.

References

[1] R. Abraham and R. Shaw. *Dynamics - The Geometry of Behaviour*. The Visual Mathematics Library, 1984.

[2] A. Appel, F. J. Rohlf, and A. J. Stein. The Haloed Line Effect for Hidden Line Elimination. In *Proc. of ACM SIGGRAPH*, pages 151–157, 1979.

[3] S. Bruckner, S. Grimm, A. Kanitsar, and E. Gröller. Illustrative Context-Preserving Exploration of Volume Data. In *Proc. of IEEE Visualization*, pages 1559–1569, 2006.

[4] C. Correa, D. Silver, and M. Chen. Illustrative Deformation for Data Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1320–1327, 2007.

[5] U. Dallmann. *Topological Structures of Threedimensional Flow Separations*. PhD thesis, Deutsches Zentrum für Luft- und Raumfahrt, 1983.

[6] P. Decaudin. Cartoon-Looking Rendering of 3D-Scenes. Technical Report 2919, INRIA Rocquencourt, June 1996.

[7] J. Diepstraten, D. Weiskopf, and T. Ertl. Transparency in Interactive Technical Illustrations. In *Proc. of Eurographics*, volume 21, pages 317–325, 2002.

[8] D. Dooley and M. F. Cohen. Automatic Illustration of 3D Geometric Models: Lines. In *ACM Symposium on Interactive 3D Graphics*, pages 77–82, New York, NY, USA, 1990. ACM.

[9] D. Dooley and M. F. Cohen. Automatic Illustration of 3D Geometric Models: Surfaces. In *Proc. of IEEE Visualization*, pages 307–314, 1990.

[10] M. Everts, H. Bekker, J. Roerdink, and T. Isenberg. Depth-Dependent Halos: Illustrative Rendering of Dense Line Data. In *Proc. of IEEE Visualization*, pages 1299–1306, 2009.

[11] J. Fischer, D. Bartz, and W. Straßer. Illustrative Display of Hidden Iso-Surface Structures. In *Proc. of IEEE Visualization*, pages 663–670, 2005.

[12] C. Garth, H. Krishnan, X. Tricoche, T. Bobach, and K. Joy. Generation of Accurate Integral Surfaces in Time-Dependent Vector Fields. In *Proc. of IEEE Visualization*, pages 1404–1411, 2008.

[13] C. Garth, X. Tricoche, T. Salzbrunn, T. Bobach, and G. Scheuermann. Surface Techniques for Vortex Visualization. In *Proc. of Eurographics/IEEE Symposium on Visualization*, pages 155–164, 2004.

[14] B. Gooch and A. Gooch. *Non-Photorealistic Rendering*. A K Peters, Natick, MA, 2001.

[15] J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, volume 42 of *Applied Mathematical Sciences*. Springer, Heidelberg, 2002. Corrected seventh printing.

[16] W. Hsu, J. Mei, C. Correa, and K. Ma. Depicting Time Evolving Flow with Illustrative Visualization Techniques. In *Proc. of International Conference on Arts and Technology*, September 2009.

[17] J. Hultquist. Interactive Numerical Flow Visualization Using Stream Surfaces. *Computing Systems in Engineering*, 1(2-4):349–353, 1990.

[18] T. Isenberg, B. Freudenberg, N. Halper, S. Schlechtweg, and T. Strothotte. A Developer's Guide to Silhouette Algorithms for Polygonal Models. *IEEE Computer Graphics and Applications*, 23(4):28–37, 2003.

[19] A. Joshi, J. Caban, P. Rheingans, and L. Sparling. Case Study on Visualizing Hurricanes Using Illustration-Inspired Techniques. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):709–718, 2009.

[20] A. Joshi and P. Rheingans. Illustration-Inspired Techniques for Visualizing Time-Varying Data. In *Proc. of IEEE Visualization*, pages 679–686, 2005.

[21] M. Kaplan. Hybrid Quantitative Invisibility. In *Proc. of Symposium on Non-Photorealistic Animation and Rendering*, pages 51–52, 2007.

[22] R. Kirby, H. Marmanis, and D. Laidlaw. Visualizing Multivalued Data from 2D Incompressible Flows Using Concepts from Painting. In *Proc. of IEEE Visualization*, pages 333–340, 1999.

[23] H. Krishnan, C. Garth, and K. Joy. Time and Streak Surfaces for Flow Visualization in Large Time-Varying Data Sets. In *Proc. of IEEE Visualization*, pages 1267–1274, 2009.

[24] J. Krüger and R. Westermann. Efficient Stipple Rendering. In *Proc. of IADIS Computer Graphics and Visualization*, 2007.

[25] R. Laramee, C. Garth, J. Schneider, and H. Hauser. Texture Advection on Stream Surfaces: A Novel Hybrid Visualization Applied to CFD Simulation Results. In *Proc. of Eurographics/IEEE Symposium on Visualization*, pages 155–162, 2006.

[26] W. Li, L. Ritter, M. Agrawala, B. Curless, and D. Salesin. Interactive Cutaway Illustrations of Complex 3D Models. In *Proc. of ACM SIGGRAPH*, number 31, 2007.

[27] H. Löffelmann, L. Mroz, E. Gröller, and W. Purgathofer. Streamarrows: Visualizing Multiple Layers of Streamsurfaces. Technical report, The Visual Computer, 1996.

[28] H. Löffelmann, L. Mroz, E. Gröller, and W. Purgathofer. Stream Arrows: Enhancing the Use of Stream Surfaces for the Visualization of Dynamical Systems. *The Visual Computer*, 13(8):359–369, 1997.

[29] T. McLoughlin, R. Laramee, R. Peikert, F. Post, and M. Chen. Over Two Decades of Integration-Based, Geometric Flow Visualization. In *Eurographics State of the Art Reports*, pages 73–92, 2009.

[30] M. Nienhaus and J. Döllner. Blueprints - Illustrating Architecture and Technical Parts Using Hardware-Accelerated Non-Photorealistic Rendering. In *Proc. of Graphics Interface*, pages 49–56, 2004.

[31] H. Obermaier, J. Kuhnert, M. Hering-Bertram, and H. Hagen. Stream Volume Segmentation of Grid-less Flow Simulation. In *Proc. of Topological Methods in Data Analysis and Visualization*, page to appear, 2004.

[32] D. Patel, C. Giertsen, J. Thurmond, J. Gjelberg, and E. Gröller. The Seismic Analyzer - Interpreting and Illustrating 2D Seismic Data. In *Proc. of IEEE Visualization*, pages 1571–1578, 2008.

[33] R. Peikert and F. Sadlo. Topologically Relevant Stream Surfaces for Flow Visualization. In *Proc. of Spring Conference on Computer Graphics*, pages 43–50, 2004.

[34] R. Peikert and F. Sadlo. Topology-guided Visualization of Constrained Vector Fields. In H. Hauser, H. Hagen, and H. Theisel, editors, *Topology-Based Methods in Visualization*, pages 21–34. Springer-Verlag, 2007.

[35] Z. Salah, D. Bartz, W. Straßer, and M. Tatagiba. Expressive Anatomical Illustration Based on Scanned Patient Data. *GMS Current Topics in Computer and Robot Assisted Surgery Journal*, 1, 2006.

[36] R. Sayeed and T.L.J.Howard. State-of-the-art of Non-Photorealistic Rendering (NPR) for Visualisation. In *Theory and Practice of Computer Graphics 2006*, 2006.

[37] T. Schafhitzel, E. Tejada, D. Weiskopf, and T. Ertl. Point-Based Stream Surfaces and Path Surfaces. In *Proc. of Graphics Interface*, pages 289–296, 2007.

[38] G. Scheuermann, T. Bobach, H. Hagen, K. Mahrous, B. Hamann, K. Joy, and W. Kollmann. A Tetrahedra-Based Stream Surface Algorithm. In *Proc. of IEEE Visualization*, pages 151–158, 2001.

[39] D. Schneider, W. Reich, A. Wiebel, and G. Scheuermann. Topology Aware Stream Surfaces. In *Proc. of Eurographics/IEEE Symposium on Visualization*, 2010.

[40] D. Schneider, A. Wiebel, and G. Scheuermann. Smooth Stream Surfaces of 4th Order Precision. In *Proc. of Eurographics/IEEE Symposium on Visualization*, pages 871–878, 2009.

[41] M. Sousa, D. Ebert, D. Stredney, and N. Svakhine. Illustrative Visualization for Medical Training. In *Proc. of EG Workshop on Computational Aesthetics in Graphics, Visualization and Imaging*, pages 201–208, 2005.

[42] D. Stalling. *Fast Texture-Based Algorithms for Vector Field Visualization*. PhD thesis, Freie Universität Berlin, 1998.

[43] T. Strothotte and S. Schlechtweg. *Non-Photorealistic Computer Graphics: Modeling, Rendering, and Animation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 2002.

[44] N. Svakhine, Y. Jang, D. Ebert, and K. Gaither. Illustration and Photography Inspired Visualization of Flows and Volumes. In *Proc. of IEEE Visualization*, pages 687–694, 2005.

[45] H. Theisel, T. Weinkauf, H. Hege, and H. Seidel. Saddle Connectors - An Approach to Visualizing the Topological Skeleton of Complex 3D Vector Fields. In *Proc. of IEEE Visualization*, pages 225–232, 2003.

[46] C. Tietjen, T. Isenberg, and B. Preim. Combining Silhouettes, Surface, and Volume Rendering for Surgery Education and Planning. In *Proc. of Eurographics/IEEE Symposium on Visualization*, pages 303–310, 2005.

[47] X. Tricoche, C. Garth, G. Kindlmann, E. Deines, G. Scheuermann, M. Ruetten, and C. Hansen. Visualization of Intricate Flow Structures for Vortex Breakdown Analysis. In *Proc. of IEEE Visualization*, pages 187–194, 2004.

[48] G. Turk and D. Banks. Image-Guided Streamline Placement. In *Proc. of ACM SIGGRAPH*, pages 453–460, 1996.

[49] I. Viola and E. Gröller. Smart Visibility in Visualization. In *Proc. of International Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*, pages 209–216, 2005.

[50] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 2004.

[51] J. Weber. ProteinShader: Illustrative Rendering of Macromolecules. *BMC Struct Biol*, 9(19), 2009.