Paper C

A Hierarchical Splitting Scheme to Reveal Insight into Highly Self-Occluded Integral Surfaces

Andrea Brambilla[1], Ivan Viola[1], Helwig Hauser[1]

[1] Department of Informatics, University of Bergen, Norway

Abstract

In flow visualization, integral surfaces are of particular interest for their ability to describe trajectories of massless particles. In areas of swirling motion, integral surfaces can become very complex and difficult to understand. Taking inspiration from traditional illustration techniques, such as cut-aways and exploded views, we propose a surface analysis tool based on surface splitting and focus+context visualization. Our surface splitting scheme is hierarchical and at every level of the hierarchy the best cut is chosen according to a surface complexity metric. In order to make the interpretation of the resulting pieces straightforward, cuts are always made along isocurves of specific flow attributes. Moreover, a degree of interest can be specified, so that the splitting procedure attempts to unveil the occluded interesting areas. Through practical examples, we show that our approach is able to overcome the lack of understanding originating from structural occlusion.

This article was published in *Journal of WSCG*, 20, 1, 57–64, 2012.

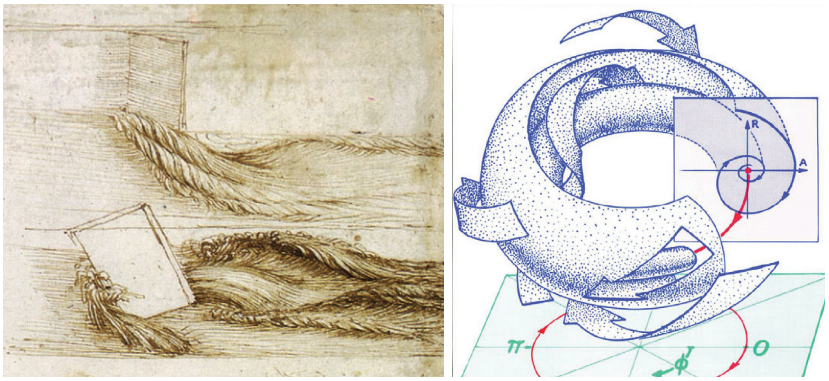

Figure 1: (left) Example of a cut-away view in a traditional illustration by Leonardo da Vinci [dV13]. (right) Illustration of a stream surface with cuts and clipping planes, by Abraham and Shaw [AS82].

1 Introduction

Flow phenomena are present at very different scales in our world, and they influence many aspects of our daily life: winds and water currents determine weather and climate, the stream of air around vehicles affects their speed and stability, the flow of blood in our vessels is fundamental for our good health condition. Understanding their behaviour is therefore highly relevant in many fields, and several years of research in *flow visualization* have produced a wide set of tools to accomplish this difficult task [PVH+02].

Flow behaviour can be analyzed from different points of view, according to the specific needs of the user. In particular, field experts are often interested in the trajectories of massless particles that are advected by the flow, which are commonly visualized using *integral curves*. Specifically, a *path line* represents the trajectory of a massless particle seeded from a specific starting location. Similarly, a *path surface* conveys the trajectories of a set of particles seeded along a $1D$ curve.

Integral surfaces are very expressive, but have a major downside: in correspondence with areas of swirling motion, like vortices and eddies, they tend to fold and twist, becoming very intricate and difficult to understand (Figures 2, 7, and 8). In this paper, we present a procedure which aims at solving this issue using techniques from traditional handcrafted illustration, such as cutting and splitting (Figure 1). These concepts have been frequently applied in medical visualization scenarios, but their application in the context of flow visualization has been limited. This is probably due to the fact that identifying well defined objects in

flow data is very challenging. An overview of related approaches is presented in Section 2.

We propose a general surface splitting methodology based on two main concepts: a *cut space* defines possible ways to split a surface so that the resulting pieces have a clear meaning, while a *complexity measure* determines a degree of occlusion at every point on the surface. We iteratively split the surface according to a cut from the cut space, so that the complexity is reduced the most. To improve the versatility of our approach, we allow the user to specify a degree of interest (DoI) function over the surface, which is combined with the complexity measure when the cut is chosen. Details on the splitting algorithm can be found in Section 3.

The resulting pieces of the surface are presented in a tree-like structure, and pieces of interest can be visualized either separated from the rest of the flow structure, or with a semi-transparent context (Figure 2). We use a stream surface extracted from the ABC flow to illustrate our method. We then show the application of our method on two datasets from application fields. Section 4 describes this process and provides a short discussion on timings and computational complexity.

Compared to the current state of the art, the main contributions of our work are:

- a general methodology for the design of surface cuts
- the first (to the best of our knowledge) splitting approach for integral surfaces
- a novel complexity measure for surfaces, which can take into account the importance of the data
- a helpful tool for the analysis of stream surfaces.

2 Related Work

According to one of the most well-known categorizations [PVH+02], flow visualization techniques can be classified in four groups: direct, texture-based, geometric and feature-based visualization. Our work is related to the third category. Geometric approaches in fact aim at visualizing flow data through *integral structures*. The most common types of $1D$ integral curves are

- *streamlines*: curves tangent to the flow field in every point at a specific time instant
- *path lines*: the trajectories of massless particles in steady or unsteady flows
- *streak lines*: formed by particles continuously released in the velocity field from a specific location

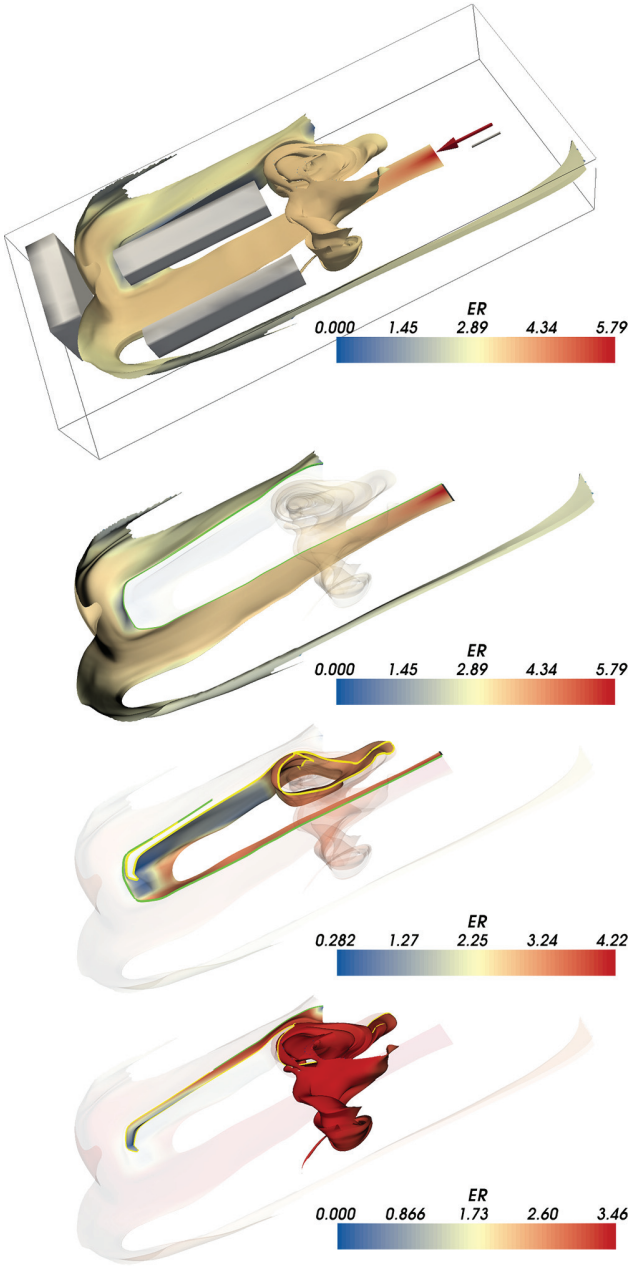

Figure 2: A stream surface extracted from a simulation of a gas leak on an oil platform. Top image: the initial surface with the position of the leak (red arrow) and the objects placed in the room (gray structures). Bottom three images: the surface pieces obtained after two cuts.

- *time lines*: curves connecting a set of particles simultaneously released along a seeding curve.

These concepts can be extended to $2D$ and $3D$, obtaining surfaces and volumes respectively. Interested readers can refer to to the excellent survey by McLoughlin et al. [MLP+10] for more details.

Flow datasets are often multidimensional, multivariate and very dense. In these cases, traditional flow visualization approaches often suffer from cluttering and occlusion problems, which are commonly addressed with simple techniques, such as clipping, slicing or conventional transparency. A novel visualization research direction, called *illustrative visualization* [RBGV08], aims at solving these perceptual issues taking inspiration from traditional handcrafted illustrations.

Cutting an object to reveal its inner parts is a common approach in illustrative visualization, and it can be applied in different ways. A typical example are *exploded views*: Li et al. [LACS08] apply this concept to show how composite objects are built. Ruiz et al. [RVB+08] suggest to subdivide a volume into oriented slabs according to the amount of information conveyed. More recently, Karpenko et al. [KLMA10] propose an explosion strategy for mathematical surfaces based on surface symmetries.

If an importance measure is defined over the data, the visualization could be guided by these values. For instance, Viola et al. [VKG05] describe a volume rendering technique which discards the low-importance (context) portions of the volume occluding the relevant ones (focus). Similarly, Bruckner and Gröller [BG06] propose an exploded view strategy, where the occluding context is not discarded, but displaced in an intuitive way. Bruckner and Gröller also presented a concise overview of basic focus+context approaches in 2005 [BG05]. An effective combination of splitting and focus+context visualization has been presented by Balabanian et al. [BVG10]. Their work is focused on medical volumetric data and the splitting is based on a precomputed segmentation. The resulting pieces are displayed in a navigable graph, which was the main inspiration for our subdivision hierarchy.

Illustrative principles have been mainly adopted in medical visualization, but, especially in recent years, they are spreading to other contexts as well. For flow visualization, a fair number of illustrative techniques have been proposed [BCP+12]. The self-occlusion problem of integral surfaces have been initially addressed in an early paper by Löffelman et al. [LMGP97]: their approach cuts away pieces of the surface, generating results similar to the illustrations by Abraham and Shaw (Figure 1, right).

Two relevant focus+context approaches have been proposed in 2005 and 2007 respectively. The Eyelet particle tracing approach [WS05] shows integral surfaces passing through a specific point of high interest. In contrast, the technique by Correa et al. [CSC07] computes a deformation of the low importance data so that the focus is not occluded. More recently, two noteworthy approaches

[HGH+10, BWF+10] propose to address the self-occlusion problem of stream surfaces through a smart use of transparency. They also adopt ad-hoc shading and texturing in order to improve depth perception and convey local flow behaviour.

Outside the context of flow visualization, similar issues have been investigated in connection with isosurfaces of scalar volumes. In this field, many techniques have been proposed (the contour spectrum [BPS97], Reeb graphs [FTAT00] and similarity maps [BM10a], just to mention a few), but their applicability to flow data is still uncertain.

3 Surface Splitting

In the case of $3D$ flow fields, a stream surface is a $2D$ manifold. Our algorithm assumes it is represented by a triangular mesh. The mesh is defined by a set of points $P \subset \mathbb{R}^3$, and a set of triangles T. Flow data is sampled at each point in P: for instance, the velocity at a point $\mathbf{p} \in P$ is $\mathbf{v}(\mathbf{p})$. Linear interpolation is used to determine flow attributes over the triangles.

The structure of our general splitting framework is summarized in Figure 2. The splitting process is iterative and begins when the user requests to generate a cut. At this point two independent steps are performed: the complexity measure

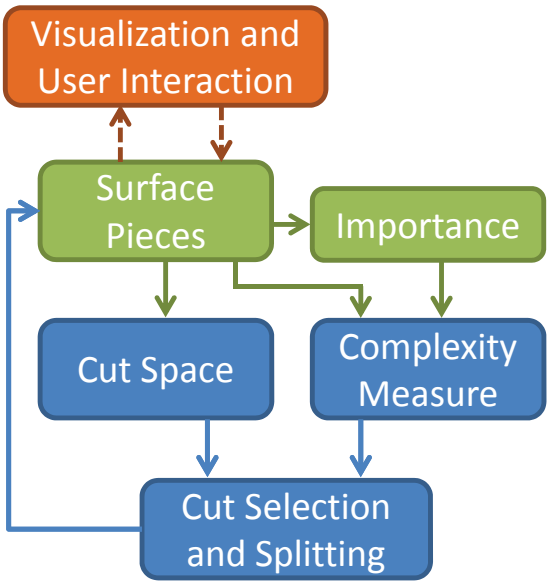

Figure 3: An overview of the splitting algorithm.

$cpx(\cdot)$ is computed for every $\mathbf{p} \in P$ and a set of potential cuts (the cut space) is generated. The complexity measure can take into account a degree of interes $doi(\cdot)$ defined over the points.

Notice that, regardless of how a cut is defined, it is always possible to reduce it to a *cutting curve* on the surface, i.e., the line along which a cut would split the surface. Therefore, for every potential cut, the complexity values are integrated along the corresponding cutting curve, and the cut with the highest overall complexity $CPX(\cdot)$ is chosen. The surface is finally split along the chosen cut, and the resulting pieces are inserted in the subdivision hierarchy (a binary tree) as children of the initial surface. The user can explore the tree and possibly request a new cut, executing again the whole procedure over all the leaves of the tree.

This is a general scheme to design effective splitting approaches, every step of the process can be customized according to the kind of surface of interest and to the desired results. In the following, we describe all the operations in detail and explain how we have tuned this framework in order to effectively split stream surfaces.

3.1 The complexity measure

The complexity measure $cpx(\cdot)$ is a function that associates a certain complexity value to every $\mathbf{p} \in P$. The meaning of this value depends on how the function is computed. Since our goal is to reduce occlusion, we define the complexity so that $cpx(\mathbf{p})$ represents how much $\mathbf{p}$ conceals the rest of the surface. However, to accurately evaluate such a measure, all the possible viewpoints should be considered, which is too expensive to allow for user interaction. We opted for an approximation based on a simple consideration: datasets are frequently shown using a polar view, with the camera moving circularly around a pivoting point $\mathbf{o}$ placed at the center of the object of interest. Thus, we consider the amount of occlusion generated by $\mathbf{p}$ when the camera is looking directly at it, i.e., when it lies exactly between the camera and the pivot. Let $\mathbf{r} = \mathbf{o} - \mathbf{p}$ be the vector from $\mathbf{p}$ to $\mathbf{o}$, we set

$$cpx(\mathbf{p}) = \|X\| \tag{1}$$

where X is the set of intersection points between $\mathbf{r}$ and the surface mesh.

There is however an issue to solve: if $\mathbf{r}$ is tangent to portions of the surface, $cpx(\mathbf{p})$ can easily degenerate (Figure 4, middle red line). To attenuate this effect, we additionally take into account the angle between $\mathbf{r}$ and the surface normals $nrm(\cdot)$ at the intersection points

$$cpx(\mathbf{p}) = \sum_{\mathbf{x} \in X} \left| \mathbf{nrm}(\mathbf{x}) \cdot \frac{\mathbf{r}}{\|\mathbf{r}\|} \right| \tag{2}$$

Including the importance measure is straightforward. We have to modify the complexity function so that, if the occluded area is highly important, the com-

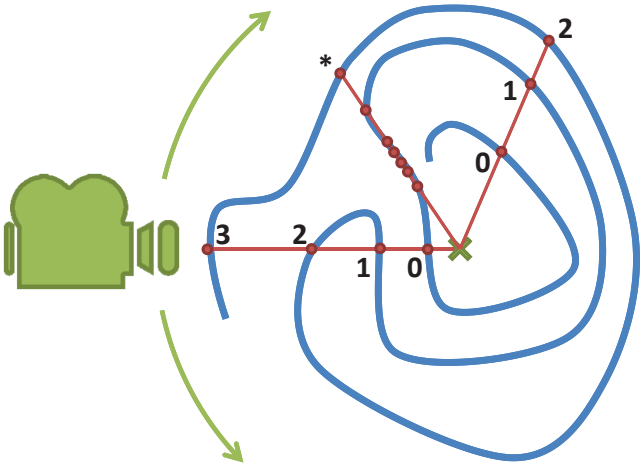

Figure 4: The typical visualization scenario. The camera (in green) moves circularly around the surface (in blue). The complexity measure, shown for a few points, is computed counting the intersections between the surface and the point-to-pivot line segment (in red).

plexity of the occluding points has to be high as well. We assume that the degree of interest function is a generic attribute $doi(\cdot)$ defined for every $\mathbf{p} \in P$:

$$cpx(\mathbf{p}) = \sum_{\mathbf{x} \in X} doi(\mathbf{x}) \left| \mathbf{nrm}(\mathbf{x}) \cdot \frac{\mathbf{r}}{\|\mathbf{r}\|} \right| \tag{3}$$

For the moment, we assume that $doi(\cdot)$ is defined at the beginning and never changes during the analysis phase; inclusion of interactive brushing techniques will be investigated in the future.

3.2 The cut space

The set of potential cuts can be defined in several ways. For example, Karpenko et al. [KLMA10] define it as a set of planes orthogonal to an explosion axis. Li et al. [LACS08], instead, define cuts as the boundaries of the components of the initial object. The fundamental requirement is that the elements of the cut space split the surface in meaningful and easily understandable pieces. In the case of flow data, defining such a space is not trivial: arbitrary cuts with a fixed geometry, such as planes or cubes, can reduce cluttering but the resulting pieces would be of difficult interpretation. Moreover, integral surfaces are not aggregate objects, so their building blocks cannot be easily defined.

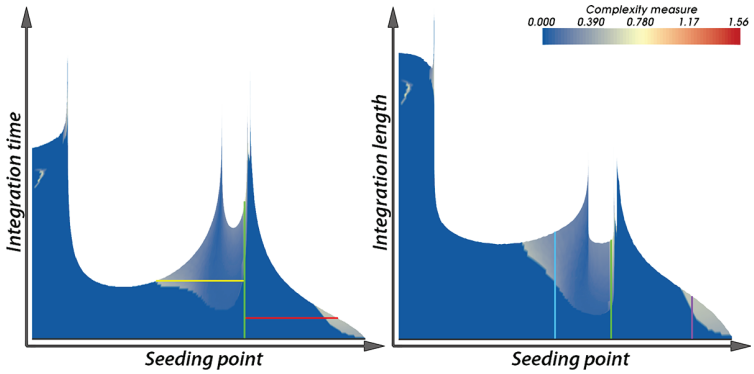

Figure 5: A stream surface from the ABC flow shown in parameter space, with three cuts. (left) parametrization given by the seeding point and the integration time. (right) The integration distance is used instead of the integration time.

One of the main characteristics of stream surfaces is that they have a semantically meaningful parametrization: every point on the surface lies in fact on the trajectory of one of the advected particles. Therefore, every point $\mathbf{p}$ can be associated with two parameters

- the *seeding point* $s(\mathbf{p})$: the location where the related particle has been seeded, expressed as a percentage of the length of the seeding line
- the *integration time* $t(\mathbf{p})$: the time needed by the related particle to travel from the seeding point to $\mathbf{p}$.

The isocurves of these two attributes are actually streamlines and time lines respectively. When a stream surface is split along one of these curves, the resulting pieces are stream surfaces as well. Therefore we define the cut space as the set of streamlines and time lines, corresponding to regular samples of their value ranges.

Notice that $s(\cdot)$ and $t(\cdot)$ are bijections. Therefore, in parameter coordinates, the surface is simply a portion of the $2D$ space, and the cuts become straight line segments parallel to the axis (Figure 5, left).

To improve the versatility of our system, we also provide the possibility of considering isocurves of arbitrary parameters. An example is shown in Figure 5, right, where the integration time has been replaced by the integration length, i.e., the arc length of the trajectory.

3.3 Surface cutting

Given the space of potential cuts, we have to determine which cut would result in the most effective reduction of structural occlusion. Recall that the complexity

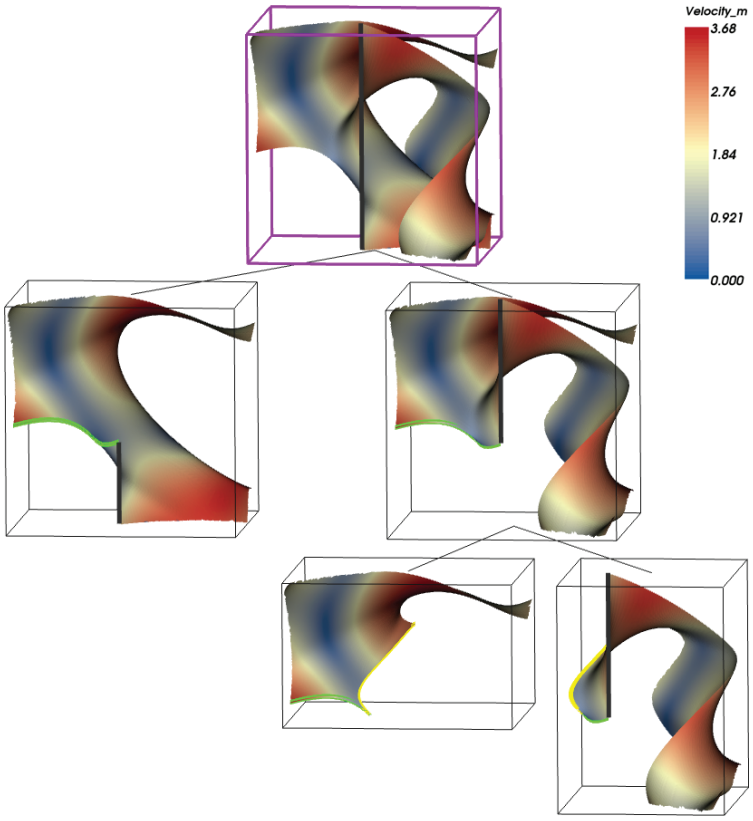

Figure 6: The tree obtained cutting two times a stream surface from the ABC flow. The first cut is made along a streamline (in green) and the second one along a time line (in yellow).

measure has been already evaluated for every point on the surface. Then, we define the overall complexity $CPX(\cdot)$ of a cut Ω as the average complexity along it:

$$CPX(\Omega) = \frac{1}{length(\Omega)} \int_{\mathbf{x}\in\Omega} cpx(\mathbf{x}) \tag{4}$$

An approximation of this integral is computed in the $2D$ parameter space as explained in Section 5.

The final step consists in selecting the cut with the highest overall complexity and using it to split the surface. However, the proposed complexity measure does not take into account the size of the resulting pieces. Usually, removing a relatively small piece from a large surface does not lead to a significant occlusion

reduction. Therefore, we bias the cut selection in two ways: firstly we discard cuts that are shorter than a specified threshold. Then we adjust the complexity of the cuts according to the area ratio of the resulting pieces.

After the optimal cut is selected, the stream surface is split and the resulting pieces are inserted in the subdivision hierarchy as children of the split surface. We never had to modify the mesh structure to get well defined cuts, but, for low resolution models, a triangle splitting procedure may be required.

Notice that, if the surface has already been subdivided, the cut evaluation is performed on all the current pieces. Then, only the piece with the highest complexity cut is split.

The subdivision hierarchy is presented to the user as in Figure 6. At every node of the tree, the corresponding surface piece is displayed. The user can interact with this view to get an overall idea of the generated cuts. Then a single piece can be selected and visualized in a separate view in a focus+context manner: the piece of interest is rendered completely opaque while the rest of the surface can be optionally shown with variable transparency, as in Figure 7, bottom row.

4 Demonstration

In order to show the capabilities of our visualization system, we used it to explore stream surfaces extracted from one synthetic and two CFD datasets. In the following, we give details about the considered datasets and discuss the most relevant results.

4.1 ABC flow

The *ABC flow* is a synthetic dataset well known in flow visualization [DFH+86]. It is defined as a vector field over the domain $[0, 2\pi]^3 \in \mathbb{R}^3$ and the velocities are given by:

$$\mathbf{v}(x, y, z) = \begin{pmatrix} A\sin(z) + B\cos(y) \\ B\sin(x) + C\cos(z) \\ C\sin(y) + A\cos(x) \end{pmatrix} \tag{5}$$

which are solutions of the Euler equation for inviscid flow. We set $A = sqrt(3)$, $B = sqrt(2)$, and $C = 1$. An overview of the dataset is given in Figure 7: the top left picture shows the boundaries of the domain and one expressive stream surface we extracted; the top right picture depicts the flow behaviour on the $z = \pi$ plane.

The stream surface under consideration has two almost overlapping areas in the bottom part, one on the left and one on the right. If we do not take into account any DoI, we expect that the splitting procedure separates these areas of the surface. That is exactly what happens after the first cut in Figure 6. The situation is even more interesting if we set the DoI proportional to the velocity

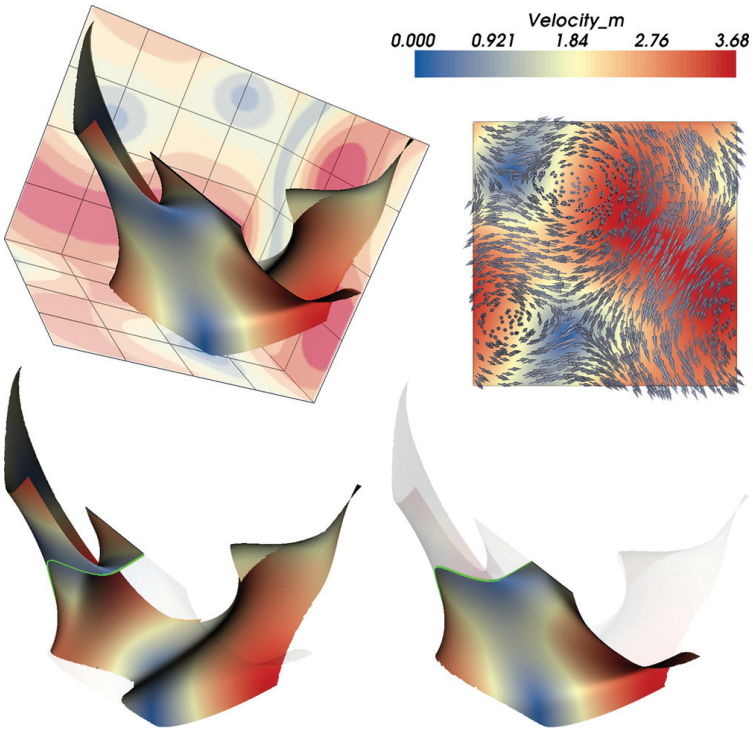

Figure 7: (top left) Overview of the ABC flow dataset, with a stream surface we extracted. (top right) A slice from the ABC flow where the velocity is depicted with glyphs. (bottom) The two pieces obtained by cutting the surface once, using the magnitude of the velocity as DoI. The complementary pieces of surface are shown semi-transparent to provide the context.

magnitude: as can be seen in Figure 7, bottom row, the first cut is made so that the high velocity areas at the bottom right are clearly visible.

4.2 Flow in a box

The second dataset we investigated using our framework is a CFD simulation of fluid flow in a box-like structure. As illustrated in Figure 8, left, the inlet is placed on the far upper side, while the outlet is situated on the front plane, adjacent to both the right and the bottom wall. Vortices and eddies are expected close to where the inlet connects to the box, so we seeded a stream surface in that area.

The surface adequately conveys the rotational behaviour, but, due to self occlusion, it is very difficult to understand what is actually happening in the inner

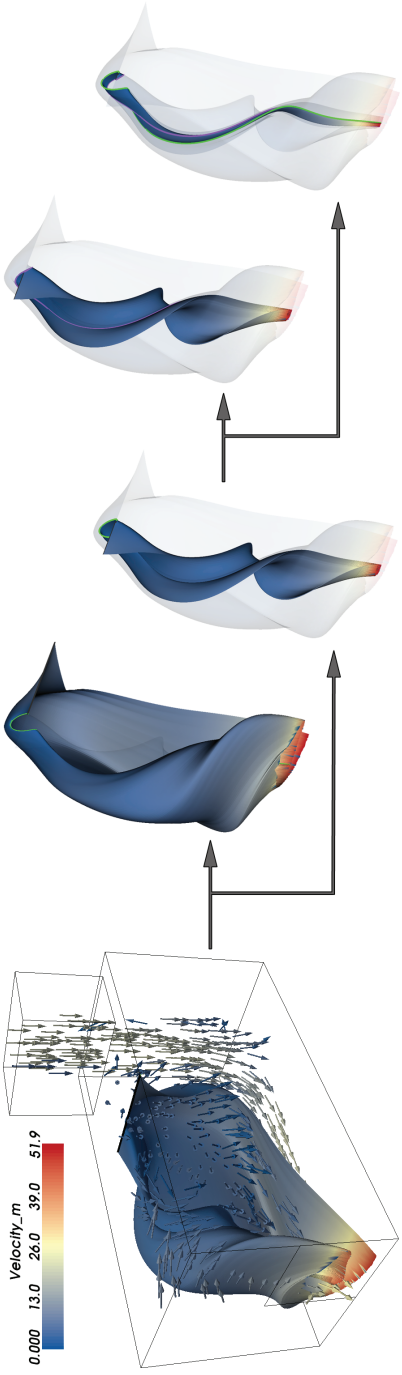

Figure 8: CFD simulation of a fluid flow in a box. The leftmost picture gives an overview of the dataset with the extracted stream surface. The other pictures show the surface after the first and the second cut.

part. After applying a first cut, the more stable piece of the surface is separated from the swirling one, effectively showing the inner vortex (Figure 8, second and third pictures from the left). Requesting an additional cut, the twisting piece is split again (Figure 8, fourth and fifth pictures). This exposes the inner part of the surface and let us analyze the swirling behaviour close to the core of the vortex. Achieving the same goals with traditional techniques, such as transparency or clipping, would have been substantially more difficult.

4.3 Gas leak simulation

The last dataset is a CFD simulation of a gas leak in a closed room on an oil platform. An overview of the architectural structure is given in Figure 2, top. The left and right walls are semi-permeable and, in normal condition, there is an almost constant flow of air in the room, from right to left. After the gas begins leaking, it mixes with air and affects the regular air flow.

The gas/air mixture is described by the *equivalence ratio* (ER), which roughly represents the ratio between fuel and oxidant. In our scenario, where ER is

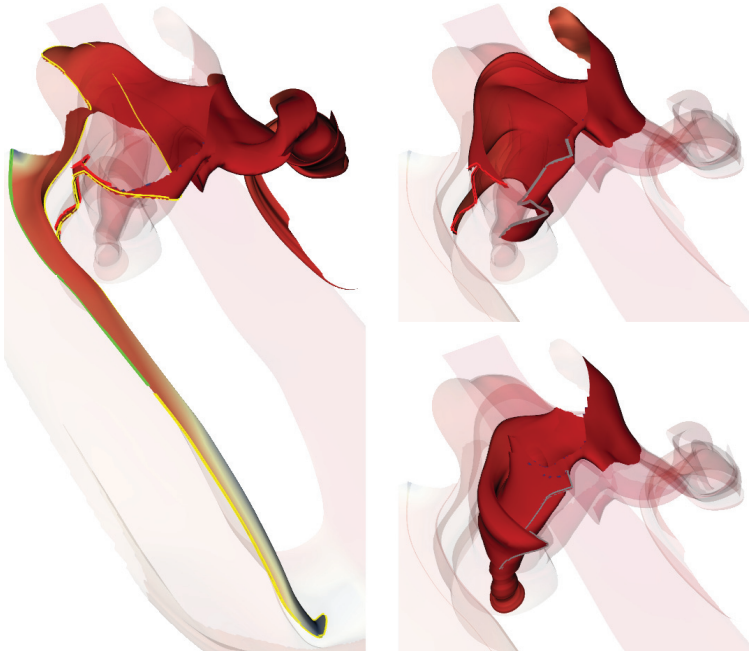

Figure 9: Pieces of stream surface extracted from the gas leak dataset. Iteratively cutting the surface with the proposed approach allows for an easy investigation of the inner areas of the vortices.

between 0.5 and 1.7 the mixture is flammable, while ER greater than 1.7 means that the mixture cannot burn but it is not breathable either. One of the aspects of interest in this dataset is identifying the locations where there is mixing between air and gas.

We seeded a stream surface in front of the gas leak and observed its behaviour. Two vortices can be easily identified in the top part of the spatial domain and, given their proximity to the leak, they may have a strong influence on the mixing process. Our splitting approach, already at the first cut, correctly separates the branch with the two vortices from the rest of the surface (Figure 2). Figure 9 shows the effect of subsequent cuts: the swirling areas of the surface are effectively subdivided, and the resulting pieces can be more easily investigated and analyzed.

We received positive feedback from a domain expert. Our splitting scheme is deemed effective in simplifying stream surfaces, easing the analysis phase. The approach is considered well suited for the validation of dispersion models and, in general, for the study of turbulence and small scale phenomena.

5 Implementation

The splitting algorithm can be briefly summarized as follows: when a cut is requested, for every current piece of the surface the complexity is computed, the cut space is generated, the best cut is identified and finally the corresponding piece is split. Notice that for every piece, the complexity, the cut space and the best cut can be stored and reused when another cut is requested. In order to maximize the efficiency of our system, the current implementation precomputes all these values for the existing pieces. Therefore, when a cut is requested, the previously computed best cut is used to split the corresponding piece of surface, then the two resulting pieces are analyzed and the next best cut is determined.

If the mesh used to represent the stream surface has a large number of vertices and triangles, determining the best cut can take a considerable time. We aim at supporting user interaction on, at least, surfaces of average size, thus, we introduced various optimizations. First of all, the computation of the complexity measure is based on a ray casting process in the three-dimensional space. This is known to be a highly expensive procedure. But we can exploit the fact that the rays we trace are always directed towards the pivot. We then compute the spherical coordinates (r, ϕ, θ) of every vertex with respect to the pivot: in the resulting spherical space, all the rays we need to trace are parallel to the r axis, which means we have one less dimension to take into account. Moreover, in this space we can use a simple quad-tree to speed up the process.

A similar idea is adopted to approximate the integration of complexity along the cuts. In the $2D$ parameter space, the surface is a flat plane and the cuts are straight lines parallel to the axis (see Section 3.2). Therefore we compute the parameter coordinates of the points and rasterize the transformed surface

Dataset	Vertices	Triangles	Complexity Measure	Best Cut Search	Splitting
ABC flow	42 050	82 329	0.379 s	0.278 s	0.094 s
Box	166 499	322 931	1.466 s	0.582 s	0.362 s
Gas leak	151 320	286 874	1.438 s	0.475 s	0.301 s

Table 1: Summary of the execution time of every step of the pipeline.

on a $n \times n$ grid. The parameter n is user specified and determines the size of the cut space. Every row and every column of the resulting image represents a possible cut: evaluating their overall complexity is now a simple image processing procedure.

The time needed to complete any of the steps of the pipeline is heavily dependent on the number of points and triangles of the mesh. This implies that, with the current implementation, the initial surface is the one that requires the most computational efforts to be analyzed. Table 1 summarizes the execution times of every step of the pipeline on the initial surface on a 2.8 GHz CPU. It is clear that the computation of the complexity measure is still the most expensive step despite the optimization. As a matter of fact, the complexity of a vertex is completely independent from the complexity of other vertices, so its computation can be easily performed on the GPU. This will be part of future developments.

6 Conclusion and Future Work

We propose a novel illustrative flow visualization algorithm which can iteratively split an integral surface while preserving its semantic meaning. The subdivision effectively reduces the structural occlusion caused by the wrapping and twisting of the surface. The resulting pieces are presented in a focus+context fashion, and the relationships between different parts of the surface are conveyed through a subdivision hierarchy. We have applied our visualization system to study one synthetic dataset and two CFD simulations, obtaining meaningful results and receiving positive feedback from a domain expert.

We have already planned a series of changes which will improve different components of our framework. As mentioned in the previous section, we plan to rework the implementation, introducing additional optimizations and executing the parallelizable operations on the GPU. Regarding the visualization, many ideas are being evaluated: e.g., the subdivision tree can be modified in order to present both the hierarchical and the adjacency information between the surface pieces. Moreover, in the focus+context view, it can be useful to show a set of selected pieces instead of just one.

In this paper we have demonstrated our approach applied to stream surfaces, but its extension to path surfaces is straightforward. We believe that the general

idea can be applied to many different kinds of surfaces once a suitable cut space has been determined.

Acknowledgements

Many thanks the anonymous reviewers for their feedback. We are grateful to Maik Schulze and Holger Theisel for providing the code for the generation of stream surfaces. Special thanks to Josué Quilliou and GexCon AS for providing the gas leak dataset. Thanks also to AVL for provinding the dataset of the flow in a box. This report has been worked out within the scope of the SemSeg project and we acknowledge the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number 226042.