Eurographics Conference on Visualization (EuroVis) 2014
H. Carr, P. Rheingans, and H. Schumann
(Guest Editors)

Volume 33 (2014), Number 3

Opacity Optimization for Surfaces

Tobias Günther[1] Maik Schulze[1] Janick Martinez Esturo[2] Christian Rössl[1] Holger Theisel[1]

[1]Visual Computing Group, University of Magdeburg [2]Max Planck Institute for Informatics, Saarbrücken

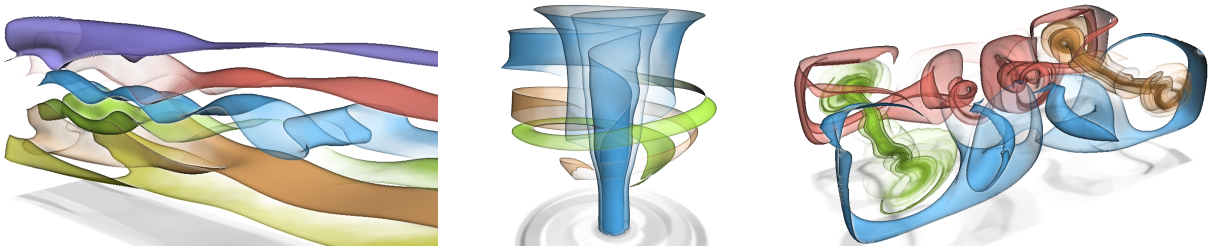


Figure 1: *Stream surfaces in a* SQUARE CYLINDER *flow, inside a* STATIC MIXER *and in a* RAYLEIGH-BÉNARD *convection (left to right). Our method fades out unimportant surfaces to clear the view on interesting structures, e.g., vortex cores. The visualizations are view-dependent, frame-coherent, at interactive rates (13–50 fps), and with 3–15 opacity updates per second.*

Abstract

In flow visualization, integral surfaces rapidly tend to expand, fold and produce vast amounts of occlusion. While silhouette enhancements and local transparency mappings proved useful for semi-transparent depictions, they still introduce visual clutter when surfaces grow more complex. An effective visualization of the flow requires a balance between the presentation of interesting surface parts and the avoidance of occlusions that hinder the view. In this paper, we extend the concept of opacity optimization to surfaces to obtain a global approach to the occlusion problem. Starting with a partition of the surfaces into patches, we compute per-patch opacity as minimizer of a bounded-variable least-squares problem. For the final rendering, opacity is interpolated on the surfaces. The resulting visualization technique is interactive, frame-coherent, view-dependent and driven by domain knowledge.

1 Introduction

Surface rendering is a classic tool to convey information in computer graphics and many other application domains. It is well-understood how to render surfaces technically and how to use sophisticated shading and rendering styles to facilitate the perception of the scene. Perception, however, is limited in the presence of occlusion to which the use of transparency is a common approach. For the rendering of transparent surfaces, visual cues such as opacity at grazing angles or silhouettes are key to convey shape. Still, multiple surfaces or even one single complex surface can result in heavily occluding cues. This may make it difficult to faithfully capture shape information from an image. In fact, this may even be impossible if occlusion hinders the view on relevant structures.

We address the occlusion problem, and we motivate this work in the context of flow exploration in scientific visualization. There, the rendering of stream lines and stream surfaces are standard tools to visualize steady vector fields. Unfortunately, their use is often limited even for simple data sets: the user is interested in understanding a volume domain that is covered by line structures or surfaces that are typically complex and experience a high degree of occlusion. This makes it difficult for the user to discover and to explore the relevant structures, e.g., vortex cores, and hence to understand the data set. Our goal is to facilitate the interactive rendering of complex surfaces with a high degree of (self-) occlusion such that important surface features and cues are visually communicated.

Recent work by Günther et al. [GRT13] solves the occlusion problem for dense line fields by a global opacity optimization. We pick up the general idea and extend it for the rendering of transparent surfaces as in Fig. 1. These automatically rendered images give a good impression of the underlying flow. For this, our method balances occlusion avoidance and an importance-driven representation of shape to provide a clear view with visual cues supporting an intuitive comprehension of the visualized shapes. We partition the surfaces into patches in a preprocess and compute at runtime for each patch an opacity by minimizing a view-dependent and frame-coherent energy that penalizes occlusions.

2 Background

Surface rendering is particularly challenging if the given shapes imply information that should be communicated best to a viewer. On the one hand, one can help the viewer to convey shape by putting emphasis on significant surface features. On the other hand, one has to avoid occlusions. Both aspects were intensively researched, as shown in Section 2.1. Our method balances both shape presentation and occlusion avoidance, based on a method explained in Section 2.2.

2.1 Illustrative Surface Rendering

Conveying Shape. Comprehensible rendering of 3D shapes has a long tradition and was pioneered by Saito and Takahashi [ST90], who extracted discontinuities, edges and contour lines in screen space. Interrante et al. [IFP96] placed curvature-oriented strokes on transparent iso-surfaces. Nienhaus and Döllner [ND04] rendered technical blueprints by peeling the layers of complex compound objects and extracted edges per layer. Luft et al. [LCD06] amplified high frequencies in the depth buffer by unsharp masking to emphasize depth discontinuities. Beside silhouettes, various classes of surface curves were suggested to enhance shape understanding like suggestive contours by DeCarlo et al. [DFRS03], apparent ridges by Judd et al. [JDA07] and Laplacian lines by Zhang et al. [ZHXC09]. To improve the perception of *transparent* surfaces, Wang et al. [WGM*08] proposed an interactive tool for color design in illustrative visualizations that respects intermixing of colors of transparent objects. Vergne et al. [VPB*09] warped environment light around surface features to enhance shape perception.

Occlusion Avoidance. Occlusion is typically avoided by transparency. Viola and Gröller [VG05] compiled a survey on smart visibility in visualization, including cutaways and ghosting, which are mainly utilized in direct volume rendering. There, Viola et al. [VKG04] adapted opacity dependent on importance. Bruckner and Gröller [BG07] improved on the perception of spatial relations by using volumetric halos. Chan et al. [CWM*09] adjusted opacity based on psychological principles, whereas Correa and Ma [CM11] used visibilty histograms to assist in transfer function design. For surfaces, Diepstraten et al. [DWE02] addressed transparency in technical illustrations by capturing styles and choices of technical artists. There exist other conceptually different ways to avoid occlusion like exploded views [APH*03] or interactive spatial separation by deformation [CSC07].

Flow Visualization. In flow visualization surfaces are used to visualize complex data [BCP*12], which has a long tradition as can be observed for the manually crafted illustrations in text books [AS84]. The rendering of integral surfaces [ELC*12], as a field of geometric flow visualization [MLP*10], was actively studied in recent years. To avoid occlusions Born et al. [BWF*10] interactively selected cuts and surface slabs, which were rendered by halftoning with additional line contours. Spencer et al. [SLCZ09] suggested the texturing of evenly-spaced streamlines on surfaces. Carnecky et al. [CSFP12] computed line integral convolution on transparent surfaces by an anisotropic diffusion of spot noise. Hummel et al. [HGH*10] enriched the surfaces by adaptive view-dependent line textures that represent tangent curves and time lines. Moreover, they proposed two transparency mappings that accentuate grazing angles. We summarize their approaches as we compare them with our method: *Angle-based transparency* maps the dot product between normal and eye vector to transparency:

$$\alpha_{\text{angle}} = \frac{2}{\pi} \arccos\langle \mathbf{n}, \mathbf{v} \rangle \; . \tag{1}$$

Normal variation considers screen space partial derivatives $\partial/\partial u$ and $\partial/\partial v$ of the view space normal's z-component:

$$\alpha_{\text{normal}} = \left((\frac{\partial n_z}{\partial u})^2 + (\frac{\partial n_z}{\partial v})^2 \right)^{\gamma/2} \; . \tag{2}$$

Carnecky et al. [CFM*13] addressed the perception of the layer order of transparent surfaces by a diffusion of silhouettes and halos that makes surface crossings distinguishable.

2.2 Opacity Optimization for Lines

The global opacity optimization method by Günther et al. [GRT13] adaptively fades out line parts such that the view is cleared on important structures. Thereby, the meaning of importance is application-dependent and allows to steer the visualization by domain knowledge. Opacity optimization minimizes an energy to compute opacity for line segments in a view-dependent, frame-coherent and *globally*-optimal way: In a preprocessing step, lines are first uniformly partitioned into n polyline segments. Then, opacity values α_i, with $i \in \{1, \ldots, n\}$, are computed for each segment as minimizer of an energy E that formalizes desired properties:

$$E = p \sum_{i=1}^{n} (\alpha_i - 1)^2 \tag{3}$$

$$+ \; q \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\alpha_i \, (1 - g_i)^{\lambda} \, h_{ij} \, g_j \right)^2 \tag{4}$$

$$+ \; r \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\alpha_i \, (1 - g_i)^{\lambda} \, h_{ji} \, g_j \right)^2 \tag{5}$$

$$+ \; s \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} (\alpha_i - \alpha_j)^2 \qquad (6)$$

with *bounded* variables $0 \leq \alpha_i \leq 1$ (0 for transparent and 1 for opaque). The first term (3) penalizes *deviation from opaqueness*, striving for as-visible-as possible primitives. By using a per-segment *importance* g_i and the *occlusion degree* h_{ij} that tells how much a segment i occludes another segment j, the second term (4) penalizes unimportant lines ($1 - g_i$ is high) in front of important ones (g_j is high). The occlusion degree h_{ij} is obtained by rasterization by counting the number of occluded fragments. Similarly, the third term (5) removes background clutter by fading out unimportant lines behind important ones. In both (4) and (5), the parameter λ steers the emphasis of important structures, i.e., the fall-off of g_i from 1. The last term (6) is a *smoothness* term that enforces low opacity variation among adjacent line segments ($a_{ij} \in \{0,1\}$). The four terms are weighted. For a normalization $p = 1$ the choices for q, r, s are discussed in [GRT13]. The minimization of E requires the numerical solution of a bounded-variable least-squares problem.

3 Problem Statement and Contribution

The most common approach to address occlusion of surfaces is to render them semi-transparently. The perception of surface order thereby resides in two key aspects: surface opacity and presentation of edges. The standard opacity mappings are *local* and accentuate grazing angles: angle-based transparency (1) and normal variation (2). When surfaces have low curvature or their boundary is not seen at a grazing angle, they are difficult to perceive. The more opaque a surface is, the better surfaces behind can be classified as such. Thus, our *first objective* is to only use transparency where strictly needed, which requires global occlusion information.

Carnecky et al. [CFM*13] applied insights from cognition to produce silhouettes that allow to deduce the order of surfaces. However, both the local transparency mapping approaches, as well as the straightforward and smart silhouette renderings tend to produce too many visual cues that occlude each other if surfaces are complex. Under certain views edges might appear that confuse the viewer, see Fig. 2. Therefore, our *second objective* is to reduce the opacity of unimportant visual cues if they hide important structures.

While the established visual cues, such as opacity at grazing angles and edge enhancements, work well for many cases, they start to fail in complex scenarios. In this paper, we propose a filtering of visual cues by transparency that is not just based on local measures, but respects the occlusion of structures and their associated user-defined importance.

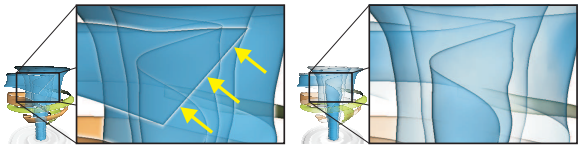
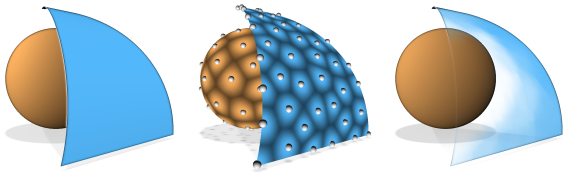


Figure 2: *Even for rather simple surfaces, silhouettes may be distracting if the object of interest is hidden. Opacity optimization: off (left) / on (right).*

4 Opacity Optimization for Surfaces

Günther et al. [GRT13] recently solved the occlusion problem for line rendering, which is summarized in Section 2.2. We seize their formulation of opacity optimization as a global optimization problem, and we reformulate it such that it can be used for surface rendering. An overview of our method is shown in Fig. 3. Starting with a partition of the initial surfaces 3(a) into patches 3(b), we compute for each patch an opacity by minimizing an energy that balances occlusion avoidance and presentation of visual cues. In our examples, cues originate from the smart transparency technique of Carnecky et al. [CFM*13]. For rendering, the resulting opacities are interpolated across the surface 3(c).



(a) Initial surfaces (b) Patches & centers (c) Result

Figure 3: *Overview: Starting with an initial set (a), we partition all surfaces into n patches (b), compute optimal opacity, which is interpolated on the surface (c). Here, $n = 100$.*

4.1 Surface Partition

In a preprocessing step, we partition the given surfaces into patches. This corresponds to the subdivision of lines into polyline segments in [GRT13], however, this step is nontrivial for surfaces. In order to solve this problem, we apply a vertex clustering based on geodesic distance. This can be interpreted as computing discrete Voronoi cells on the manifold surfaces. Assume we are given a set of seed vertices on the surface that act as centers of cells or equally as cluster representatives. We compute the approximate geodesic distance from each center to all other non-center vertices, and we assign each of these vertices to the cell whose center is nearest. Fortunately, the computation of highly accurate approximate geodesic distances can be done very efficiently by reformulation as a discrete heat-flow problem [CWW13]: This requires only the factorization of two sparse symmetric positive definite linear operators as the most expensive step. After that the computation of *all* distances can be done by backsubstitution. We use a farthest point seeding to place the centers. Starting from a randomly chosen vertex, we iterate the following step: place a new center at a vertex that has maximum geodesic distance to all existing centers. The iteration stops if the partition is fine enough, i.e., the cell size is below a threshold or a certain number of cells is reached.

The partition based on geodesic distance is reasonable as

it tends to produce compact, "convex" cells with a good ratio of circumference to area. We observe that the partition by greedy farthest point seeding is, even though it is not perfectly regular, good enough for the purpose. This means that we can avoid global patch rearrangement, e.g., by a Lloyd relaxation [Llo82], which has a slow convergence.

4.2 Averaging and Interpolation

Given is a set of surface meshes with vertices $\mathcal{V}$. The partition of the surfaces is given as a map $C : \mathcal{V} \to \mathcal{C}$ where $\mathcal{C} \subset \mathcal{V}$ denotes the cell centers, which in turn represent patches. We use the preimage $C^{-1}(i) = \{\ell \in \mathcal{V} \,|\, C(\ell) = i\}$ to describe the set of all vertices in a cell. In the remainder of this section, we use indices i, j for centers and ℓ for general vertices. For the subsequent steps, we require operators for averaging and interpolation of values.

The first one is straightforward. Given some quantity x that is defined per vertex (denoted by x_ℓ), we define

$$\text{avg}_x(i) \;:=\; |C^{-1}(i)|^{-1} \sum_{\ell \in C^{-1}(i)} x_\ell \;,$$

where $|\cdot|$ denotes the cardinality of a set.

By *interpolation* we refer to computing new values at arbitrary vertices from values given at centers. This is a bivariate scattered data interpolation problem with samples distributed on a manifold. Finding a "high quality", smooth interpolating function is a nontrivial problem that is possibly computationally expensive, e.g., requires the solution of a linear system. For our application, we require first of all a highly efficient interpolation scheme. Thus, we locally apply Shepard's method (see, e.g., [HL93]), which is a kind of inverse distance weighting. Distances $d_g(\ell, i)$ from vertex ℓ to center i are the geodesic distances (see previous section). Given any vertex $\ell \in \mathcal{V}$ we define $\mathcal{N}_k(\ell) \subseteq \mathcal{C}$ as its k nearest centers w.r.t. d_g. To evaluate a quantity x we use the subset $\{x_i \,|\, i \in \mathcal{N}_k(\ell)\}$ of the values given at centers to define

$$\text{interp}_x(\ell, k) := \frac{\sum_{i \in \mathcal{N}_k(\ell)} w_i x_i}{\sum_{i \in \mathcal{N}_k(\ell)} w_i} \quad \text{with} \quad w_i = d_g(\ell, i)^{-2} \;.$$

with $\text{interp}_x(i, k) = x_i$ for centers $i \in \mathcal{C}$. The function $\text{interp}_x(\ell, k)$ provides interpolation of values at centers with constant precision but without any theoretical guarantees on smoothness and continuity when evaluated, e.g., for adjacent vertices. This is sufficient for our purpose, even though it is not a high quality interpolation. The main advantage is that this simple scheme can be evaluated efficiently. Its overall smoothness can, to some extent, be steered by the exponent for the inverse distance; we found that weights d_g^{-2} provided best results. Here we use $k = 3$, which is small for typical uses of Shepard's method. We do this because first, we found that in our setting the quality of interpolation is still good enough. Fig. 4 shows the quality of an interpolation of mean curvature that was averaged at centers for partitions of decreasing number of patches n and $k = 3$. Second, this allows to store the indices of nearest centers simply as a 3-vector, i.e., $\mathcal{N}_3(\cdot)$ is a coordinate look-up, which enables efficient interpolation on the GPU. We precompute nearest centers and interpolation weights once in a preprocess.

4.3 Optimization

Our surface opacity optimization is based on opacity optimization for 3D line fields [GRT13], which is reviewed in Section 2.2. The main differences consist in the use of surface patches instead of polyline segments and a modification of the energy functions. In particular, the coefficients g_i, h_{ij} and a_{ij} are computed differently. Note that all sums in the energy function now iterate over patches or their centers $i, j \in \mathcal{C}$, respectively. We abstain from a formal redefinition.

Change of Energy. Opacity is computed for discrete elements. In the surface case, these are patches that result from partitioning. All quantities are measured on patches: patch *importance* g_i is an average of the importance values prescribed per vertex, for which choices are discussed in Section 4.4. For all examples, we use $g_i = avg_H(i)$, where H is the mean curvature estimated at every vertex. The *occlusion degree* h_{ij} is determined from rasterization on fragment level, which is detailed below in Section 4.5. We use a different notion of *adjacency* than for lines: The coefficients $a_{ij} > 0$ that contribute to the smoothness term (6) are now no more either 0 or 1 but continuous values that reflect geodesic distance. Thereby, centers closer to each other receive higher penalty if their opacity differs. We define for centers $i, j \in \mathcal{C}$

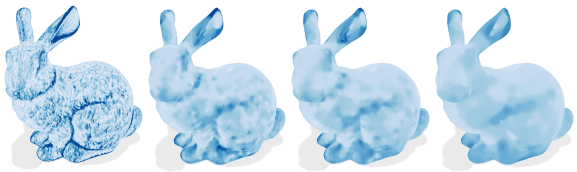
$$a_{ij} \;:=\; \frac{d_g(i,j)^{-2}}{\sum_{k \in \mathcal{N}_4(i) \setminus \{i\}} d_g(i,k)^{-2}} \;,$$

with the convention that $a_{ij} = 0$ whenever i is not within the four nearest neighbors of i, which can be interpreted as the denominator approaching infinity. Note that $i \in \mathcal{N}_4(i)$, since i is the closest center to itself. Thus, 3 centers are looked up.

Energy term (3) penalizes deviation from opaqueness. As we generally render transparent surfaces, we replace it by

$$p \sum_{i \in \mathcal{C}} (\alpha_i - t)^2 \;, \tag{7}$$

which includes a user-specified *target opacity* $0 < t \leq 1$. The modified energy is still frame-coherent and leads to a least-squares problem with unknown opacity values bounded in $[0, 1]$, as described in [GRT13]. We use the same algorithm for the numerical solution.



per vertex $n = 2000$ $n = 1000$ $n = 400$

Figure 4: *Mean curvature is estimated per vertex, then averaged per patch and interpolated for n patches.*

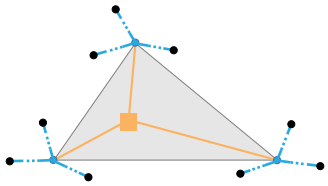


Figure 5: *Occlusion degrees h_{ij} are obtained by rasterization: The contribution of centers (•) to the rasterized fragment's (■) opacity is reconstructed. For each vertex (•) of the rasterized triangle the $k = 3$ closest centers are considered. (For illustration, their distances are shortened.) The center-to-fragment contribution is the product of interpolation weight (—) and barycentric weight (—).*

Evaluation of Opacity. Finally, the computed optimal opacity values a_i at centers $i \in \mathcal{C}$ are interpolated within patches such that every vertex $\ell \in \mathcal{V}$ is assigned an opacity

$$\alpha_\ell \;=\; \mathrm{interp}_\alpha(\ell, 3)\;.$$

At rasterization, opacity is linearly interpolated in triangles, resulting in a fragment opacity. As silhouettes are key to understand surface ordering, we apply a transfer function to the opacity of silhouette fragments to fade them out slower

$$\alpha \leftarrow 1 - (1-\alpha)^\eta \tag{8}$$

Increasing η keeps silhouettes longer, see later Section 5.2.

4.4 Importance Function

The importance g_i indicates surface parts worth to look at. In fact, the success of our method depends on its correlation to relevant structures. It might originate in local geometric information, might be sampled from scalar fields, or might be global information such as distance to a tumor. Consequentially, importance is strongly application-dependent and driveable by domain knowledge. However, simple geometric measures exist that work well in many cases, e.g., in all examples, we used mean curvature. Note that for $g_i = 1$ surface parts do not vanish since Eqs. (4) and (5) evaluate to zero.

4.5 Computing Occlusion Degree on Fragment Level

The coefficients h_{ij} in the energy term measure the occlusion that patch i causes by occluding another patch j. Similarly to [GRT13], we determine the occlusion after rasterization on fragment level. For each fragment we look up the centers and interpolation weights used for opacity interpolation. These are the $k = 3$ nearest centers for each vertex of the rasterized triangle and thus $3k$ centers in total per fragment (see Fig. 5). We maintain for every pixel a sorted linked list of the rasterized fragments [YHGT10], from which we obtain all "occluder-occludee" fragment pairs. Then we identify for every fragment pair the $3k$ centers $F \subset \mathcal{C}$ of the fragment in *front* as well as for every fragment *behind* the respective centers $B \subset \mathcal{C}$. Every fragment pair introduces a small amount of occlusion for the $3k \times 3k$ front-back center pairs i, j, with $i \in F$ and $j \in B$. The amount of occlusion is the product of the contribution of i to the occluder fragment and of j to the occluded fragment's opacity. For deducing the contribution of a center to a fragment, we store for each fragment the primitive ID (generated by the graphics runtime) and the barycentric coordinate of the fragment in the rasterized triangle. Based on the primitive ID, we look up contributing vertices, their k nearest centers and the interpolation weights. These weights are w.r.t. the triangle vertices, and they are multiplied by the barycentric weights to get per-fragment occlusion degrees. The total occlusion degree h_{ij} is determined as the sum of all contributions to fragments. We do not penalize self-occlusion of patches to prevent fading at grazing angles, i.e., we set $h_{ii} = 0$.

5 Usage and Parameters

This section elaborates on the choice of surfaces for visualizing flow data, the selection of parameters and their effect, as well as implementation details on rendering and solving.

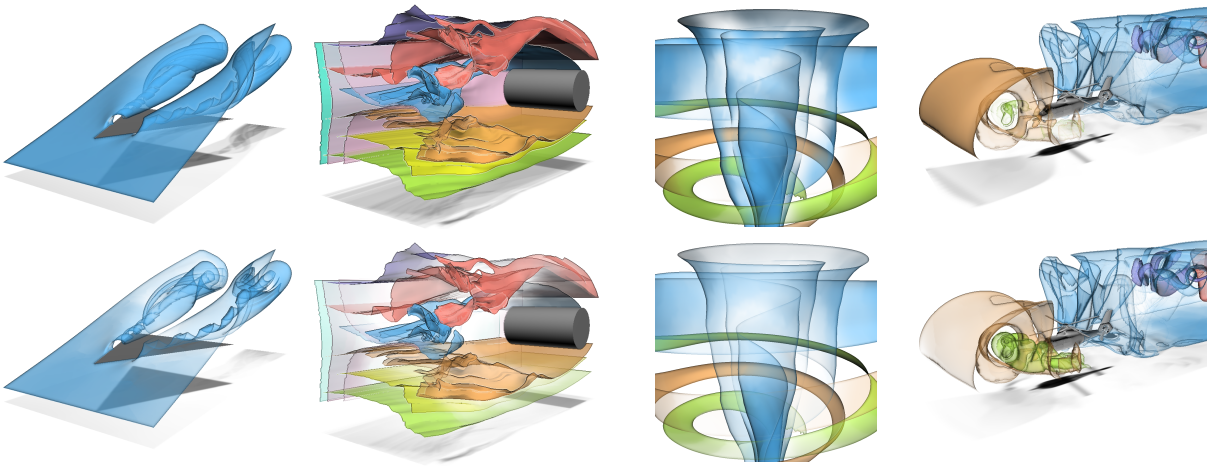
5.1 Input Surface Set

Opacity optimization only reveals structures that are contained in the set, thus the input surface set should be selected carefully. For the examples in this paper, we used Hultquist's algorithm [Hul92] for extraction and placed the seed curve manually. This can be facilitated by dragging flow-aligning surfaces [MSRT13b] for a more intuitive interaction.

There are methods for an automatic placement that could be used as well, e.g., Martinez Esturo et al. [MSRT13a] extract *one single* representative surface that is globally optimal w.r.t. a quality measure. Schulze et al. [SMG*14] and Edmunds et al. [ELM*12] present methods that produce *multiple* characteristic stream surfaces automatically, which is suitable for an exploration task.

5.2 Parameter Selection

We inherit the scene-dependent parameters introduced by [GRT13]. We discuss these parameters in the following. First, the total number of surface patches n is chosen. A higher number of patches leads to a more local adaptation of the surface opacity and thus preserves details, however, at the cost of higher solving timings. Fig. 6 depicts different choices for n. Most often, we used $n = 400$, cf. Table 1. The effect of parameters in the energy function E (see Section 2.2) are shown in Fig. 7: The weight q of the occlusion term (4) adjusts the overall opacity in the image. In Fig. 7 (left), a surface layer is removed to clear the view on the wake turbulences of a DELTA WING. The weight r of term (5) steers the removal of background clutter. In our example, the laminar stream surfaces in the background are removed to increase the visibility of the wake turbulence behind a WALL-MOUNTED CYLINDER. The weight s of the smoothness term (6) is used to enforce continuity of the surface opacity, as shown for the STATIC MIXER data set. If not

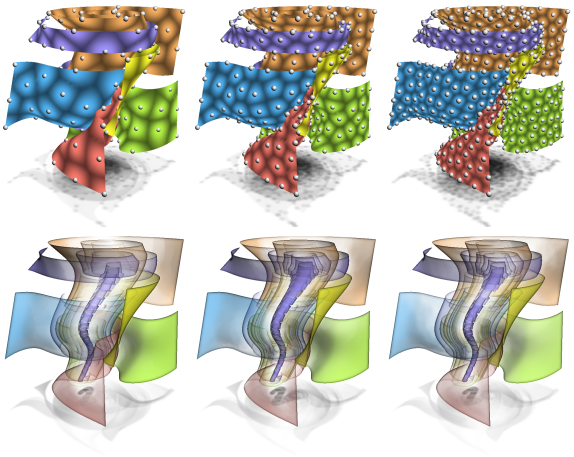


Occlusion term (4), top: $q = 0$, bottom: $q = 0.2$ | Background clutter term (5), top: $r = 0$, bottom: $r = 0.3$ | Smoothness term (6), top: $s = 0$, bottom: $s = 50$ | Emphasis exponent λ (4)&(5), top: $\lambda = 1$, bottom: $\lambda = 10$

Figure 7: *Study of the parameters inherited from opacity optimization [GRT13]*

mentioned otherwise, we set $s = 50$. Finally, the emphasis exponent λ in (4) and (5) stresses important structures: Fig. 7 (right) depicts the aerodynamics around a HELICOPTER in ground effect, studied for brown-out conditions. Of particular interest is the green vortex in front of the helicopter, as it entrains sand particles that hinder the pilot's view.

In Section 4.3, we introduced two additional parameters both with a recommended default value. First, the target opacity t in (7) is used to let surfaces strive for being *almost*, but not completely, opaque. A high value proved useful, as it supports the perception of layer order. We recommend $t = 0.9$, which is used in all examples. Second, silhouette opacity is steered by parameter η, introduced in (8), to control how fast silhouettes fade out. The effect is demonstrated in Fig. 8. As edges are helpful to hint how nearly invisible surfaces pass in front of important structures, a high value is recommended. In all our examples, $\eta = 10$ is used.



$n = 400$, solve: 18 *ms* | $n = 1000$, solve: 44 *ms* | $n = 2000$, solve: 105 *ms*

Figure 6: *Solve rates for varying number of patches n. Using a higher number of patches preserves details, though $n = 400$ already proved useful.*

6 Details on Rendering and Solving

Similar to [GRT13], we execute rendering and solving in different threads to enable an interactive response. We use fragment linked lists [YHGT10] for both the rendering of transparent surfaces (cf. Maule et al. [MCTB11]) and the extraction of occlusion degrees. The latter is done in the solver thread on the CPU, thus fragment linked lists are copied to RAM. We construct two separate lists, one for rendering and one at quarter resolution for computing occlusion degrees. The solver thread assembles the system matrix and right-hand-side and solves the system as described in Section 4.3. The rendering thread repeatedly executes two steps:

1 Render surfaces at low resolution (for computing h_{ij})
   a. Create and sort fragment linked lists
   b. Stream lists to RAM and pass to solver thread
2 Render surfaces on full resolution
   a. Upload new opacities (if solver has new solution)
   b. Create and sort fragment linked lists
   c. Establish adjacency pointers (for diffusion)
   d. Inject black silhouette and white halo intensities
   e. Diffuse silhouettes and halos (we used 5 iterations)
   f. Blend fragment lists front-to-back
   g. Render shadow on ground plane using [JB10]

Opacity optimization and the silhouette extraction by Carnecky et al. [CFM*13] (steps 2c-2e) are easily combined as they are both based on fragment linked lists. As a visual hint for the layer order, Carnecky et al. inject and diffuse white and black color at silhouettes, see Fig. 9. We modulate the injected intensity by the opacity of the fragment that

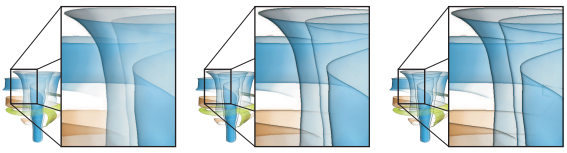


Figure 8: *Silhouette edges depend on the surface opacity, and can be longer conserved by an η-exponent. Left to right: $\eta = 1$ (neutral), $\eta = 10$ (recommended), $\eta = 50$.*

caused the silhouette. For the white halos, we require in addition to the adjacent fragments access to the predecessor of a fragment to lookup the occluder's opacity. Thus, we create doubly-linked lists in (2b).

7 Results and Discussion

Fig. 10 compares our global method with established local approaches. These are (from left to right): constant opacity, angle-based transparency and normal variation [HGH*10] (see (1) and (2), resp.), and smart transparency [CFM*13], i.e., constant opacity adapted to the number of layers with silhouettes and halos to visualize surface layer order. The last column depicts results of our surface opacity optimization, for which we used mean curvature as importance.

The first row shows stream surfaces in a STATIC MIXER. Here, the central vortex core is of interest (blue surface). Opacity optimization removes distracting surface parts, e.g., the blue part pointed out in Fig. 2, as well as the orange and green surface parts in front of and behind the core. This yields a clearer view of the interesting region.

The second row depicts a RAYLEIGH-BÉNARD CONVECTION, simulated using the free flow solver NaSt3DGP [GDN98]. The convection cells are formed by heating a thin layer of liquid from below. Among the local methods, especially normal variation (2) excels in outlining contours of curved surfaces. Due to its low opacity the perception of the layer order is difficult. This is better preserved by angle-based transparency (1) and by [CFM*13], though at the cost of occlusions. Opacity optimization removes the occlusions and clears the view on the four convection cells.

The third row shows a synthetic TORNADO data set. Here, the vortex core is of interest, which is hidden behind a number of space-filling surfaces that represent the outer flow

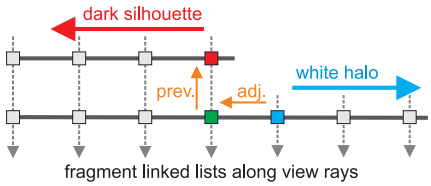


Figure 9: *Black silhouette gradients (◀–) and white halo gradients (–▶) are injected at the occluder fragment (■) and next (■) to the occluded fragment (■), see [CFM*13] for details. We modulate their intensity (which is then diffused) by the opacity of the occluder fragment (■).*

region. As seen in the center, normal variation (2) cannot reproduce planar surfaces faithfully. Surface opacity optimization performed best in accentuating the vortex core and maintaining context by removal of surface parts in front and behind the vortex core – the latter producing halos.

In the fourth row, a RAYLEIGH-BÉNARD CONVECTION is viewed from inside – a scenario that is very useful for interactive exploration. The orange surface in the foreground hinders the view on the cells, depicted in blue and green. The removal of unimportant occluding surface parts is not addressed by local approaches. Therefore, our global surface opacity optimization performed best and removed the occlusions. We see our method best suited for interactive exploration scenarios with free camera navigation.

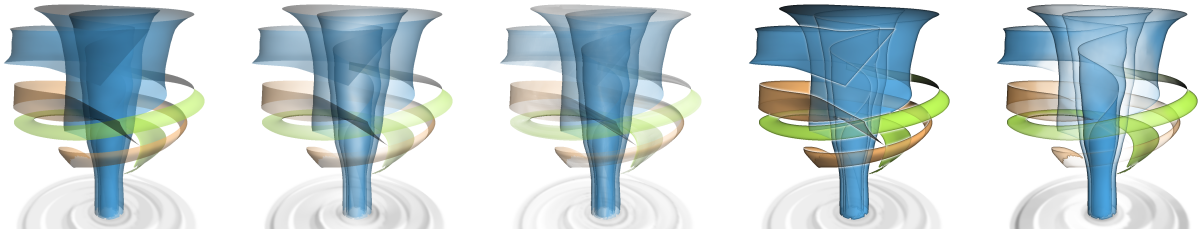
The fifth row illustrates slow blood flow in an ANEURYSM that results in highly folded and complex surfaces. All local approaches extract a high number of visual cues. Both, angle-based transparency (1) and normal variation (2), show important cues but cannot hint layer ordering. Carnecky's method [CFM*13] produces complex silhouettes that are hard to comprehend due to their sheer number. Our method removes visual cues to reveal the twisting surfaces in the inflow (orange) and the vortex in the aneurysm (blue).

The last row shows a single stream surface approaching a DELTA WING, which is provided by Markus Rütten. It starts with a laminar flow, which is problematic for normal variation (2), and then begins to fold as wake turbulences detach. Even for simple surfaces, such as these, surface opacity optimization adds additional benefit by removing the topmost layer, clearing the view on the vortex cores.

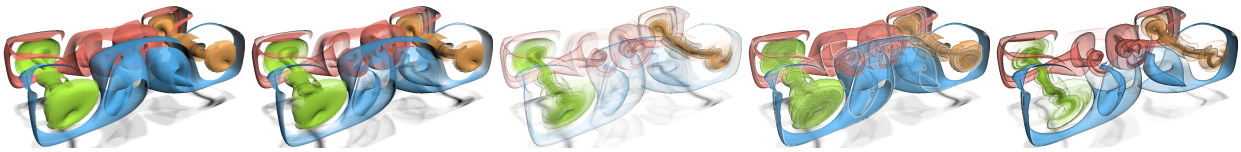
Fig. 1 (left) shows stream surfaces in the flow around a SQUARE CYLINDER. The uniformly resampled version of this vector field was provided by Tino Weinkauf and is based on the simulation by Camarri et al. [CSBI05]. The second column of Fig. 7 displays stream surfaces in the flow around a WALL-MOUNTED CYLINDER, provided by Frederich et al. [FWT08]. Distracting, laminar layers in the background are effectively removed by surface opacity optimization, leading to more visual clarity. The last column of Fig. 7 shows a HELICOPTER in forward flight close to the ground. This simulation by Kutz et al. [KKKK12] was used to study brown-out, i.e., the entrainment of sand particles by aerodynamic uplift and impacting particles. Brown-out clouds hinder the pilot's view and are mainly introduced by a vortex in front of the helicopter that is shown by the green surface.

7.1 Performance

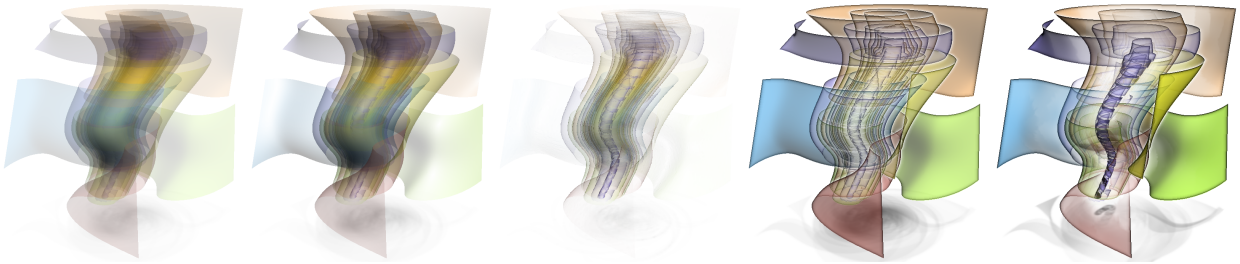
We measured the performance of our system with an Intel Core i7-2600K CPU with 3.4 GHz and Nvidia GeForce GTX 560 Ti GPU with 2 GB VRAM. All images were rendered with Direct3D at a resolution of 1000×1000 pixels. Table 1 summarizes timings for preprocessing (surface partition), rendering and solving. Column labels refer to algorithm steps in Section 6. The shadow map used for Fourier



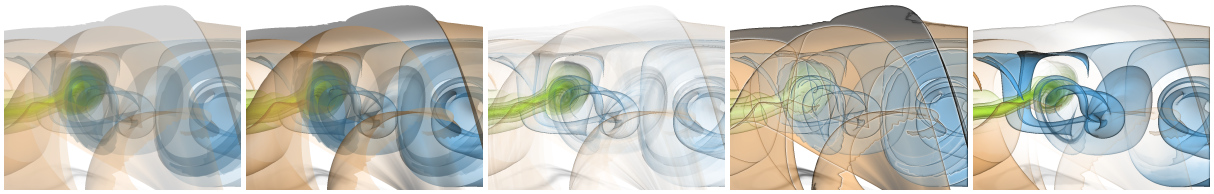
(a) Stream surfaces in a mixer ($q = 0.2$, $r = 0.25$, $\lambda = 50$).



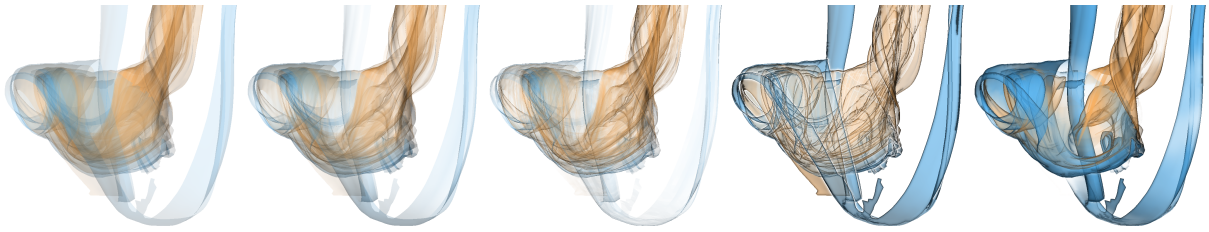
(b) Stream surfaces in a Rayleigh-Bénard convection ($q = 0.4$, $r = 0.04$, $\lambda = 8$).



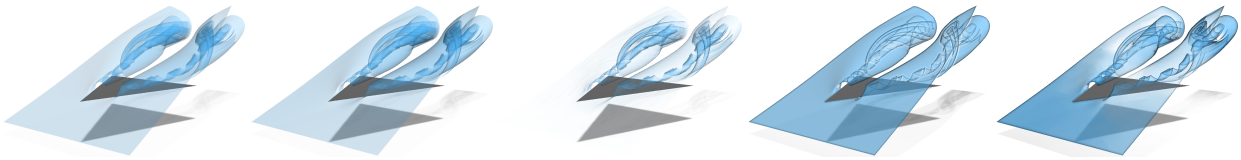
(c) Stream surfaces in a tornado ($q = 0.35$, $r = 0.16$, $s = 20$, $\lambda = 15$).



(d) Stream surfaces in a Rayleigh-Bénard convection, viewed from inside ($q = 0.03$, $r = 0.02$, $\lambda = 10$).



(e) Stream surfaces in a blood vessel with an aneurysm ($q = 0.4$, $r = 0.6$, $s = 5$, $\lambda = 11$).



(f) A single highly folded stream surface approaching a delta wing ($q = 0.54$, $r = 0.12$, $\lambda = 7$).

Figure 10: *Comparison with standard techniques, from left to right: constant transparency, angle-based [HGH*10], normal variation [HGH*10], smart transparency [CFM*13], and our novel surface opacity optimization.*

Table 1: *Performance analysis with time in milliseconds (partition in secs.). Column labels refer to algorithm steps, see Section 6. The low resolution and high resolution pass are executed in the rendering thread. The solver thread comprises assembling of the system matrix (3a) and solving (3b). The asynchronous streaming of fragment linked lists is the bottleneck (**bold**).*

		Preprocessing			Low Res. Pass		Full Resolution Pass					Solver	
Data set	**Figs.**	**Partition**	**Tris.**	n	**1a**	**1b**	**2a**	**2b**	**2c–e**	**2f**	**2g**	**3a**	**3b**
Aneurysm	10(e)	271 s	454 k	400	2.24	**206.6**	0.40	9.53	49.40	1.94	5.03	7.65	6.02
Bénard (in)	10(d)	97 s	246 k	400	1.53	**198.7**	0.26	9.81	48.82	1.95	2.08	4.70	4.62
Bénard (out)	10(b)	97 s	246 k	400	1.36	**168.2**	0.26	6.61	43.80	1.49	2.08	5.68	4.69
Delta Wing	10(f)	182 s	580 k	200	2.53	**91.2**	0.48	7.88	16.19	0.87	6.37	3.04	3.25
Helicopter	7	260 s	484 k	400	2.27	**117.7**	0.42	7.29	22.89	1.02	5.33	5.25	5.02
Square Cyl.	1	48 s	155 k	400	0.69	**53.9**	0.20	2.85	10.94	0.60	2.33	3.13	4.06
Static Mixer	10(a)	4.7 s	33 k	200	0.21	**98.8**	0.12	4.30	24.77	0.99	0.35	3.44	3.57
Tornado	10(c)	220 s	438 k	400	2.37	**194.7**	0.40	9.25	45.30	2.31	4.82	17.8	9.29
Wall. Cyl.	7	62 s	177 k	400	0.92	**115.5**	0.22	4.83	25.46	1.15	3.23	6.62	6.40

opacity mapping [JB10] in step (2f) has a pixel resolution of 512×512 and is filtered by a separated 7×7 Gaussian kernel.

The bottleneck of the system is the time required for streaming the fragment linked lists to RAM. This is because streaming is two frames deferred after enqueueing draw calls in the command buffer. Therefore, the streaming time depends on the rendering time per frame. The solving time was in our examples generally lower than the streaming time. Since both run asynchronously, we could have used more patches without loss of performance. The key to faster opacity updates therefore resides in optimizing the rendering. Figure 11 compares the performance of enhanced silhouettes of Carnecky et al. [CFM*13] with simple silhouettes, obtained by detecting depth discontinuities in image space. The latter is twice as fast in both rendering and solving, though at lower visual quality. One way to improve the scalability of the system is to use the simple silhouette detection or fewer diffusion iterations in [CFM*13] during interactive navigation and to switch to a high quality parameter set whenever the camera stops moving.

In summary, our rendering achieves interactive rates at about 13–50 frames per second, depending on the data set and the view. Due to the streaming bottleneck, opacity solutions are updated only 3–15 times per second. This latency is hidden similarly as in [GRT13] by slowly fading toward the latest opacity solution.

7.2 Limitations

Surface opacity optimization removes unimportant surface parts that occlude important scene elements. This requires the presence of visual clutter in the first place, which is only the case if both unimportant and important regions are present in the scene. If too many surfaces of likewise high importance occlude each other, it is impossible to decide, which surface should be shown. Similarly, surfaces with equally low importance, such as parallel planar surfaces, have no distinctive feature to accentuate. For such scenes, users are required to supply the importance terms with additional information to decide upon relevance. Besides, important structures might occlude each other, thus a notification of such an event is a possible step for future work. Furthermore, the results depend on the initial surface set, as opacity optimization only reveals information contained in the set.

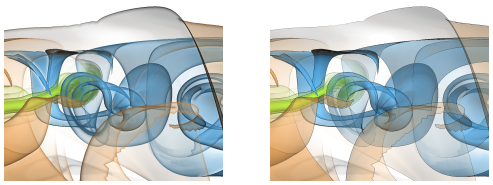


Figure 11: *The performance of the silhouette rendering has great impact on the frames and solves per second. Left: enhanced silhouettes [CFM*13] (12.8 fps, 3.8 sol/s), right: silhouettes by depth discontinuity (25.8 fps, 7.5 sol/s).*

8 Conclusions

In this paper, we addressed the occlusion problem for surfaces and developed the first global approach that determines a surface opacity that balances occlusion avoidance versus importance-driven shape representation. For this, we fade out surface parts in a view-dependent and frame-coherent way so that important structures are no longer occluded. By decoupling rendering and solving, we obtain interactive rates. Our method targets specifically scenes with multiple surfaces or highly complex scenes. For such, improvements over existing local methods are significant.

Future work includes the extension to animated surfaces and accelerated solving by partitioning the surfaces in a view-dependent way in the spirit of [GRT14]. Surface opacity optimization can also be extended to point sets, since geodesics can be computed for them, too [CWW13].

References

[APH*03] AGRAWALA M., PHAN D., HEISER J., HAYMAKER J., KLINGNER J., HANRAHAN P., TVERSKY B.: Designing effective step-by-step assembly instructions. *ACM Trans. Graph. (Proc. SIGGRAPH) 22*, 3 (2003), 828–837. 2

[AS84] ABRAHAM R. H., SHAW C. D.: *Dynamics – The Geometry of Behaviour*. The Visual Mathematics Library. Aerial Press, Incorporated, 1984. 2

[BCP*12] BRAMBILLA A., CARNECKY R., PEIKERT R., VIOLA I., HAUSER H.: Illustrative flow visualization: State of the art, trends and challenges. In *EG - STAR* (2012), pp. 75–94. 2

[BG07] BRUCKNER S., GRÖLLER E.: Enhancing depth-perception with flexible volumetric halos. *IEEE TVCG (Proc. Vis) 13*, 6 (2007), 1344–1351. 2

[BWF*10] BORN S., WIEBEL A., FRIEDRICH J., SCHEUERMANN G., BARTZ D.: Illustrative stream surfaces. *IEEE TVCG (Proc. Vis) 16*, 6 (2010), 1329–1338. 2

[CFM*13] CARNECKY R., FUCHS R., MEHL S., JANG Y., PEIKERT R.: Smart transparency for illustrative visualization of complex flow surfaces. *IEEE TVCG 19*, 5 (2013), 838–851. 2, 3, 6, 7, 8, 9

[CM11] CORREA C. D., MA K.-L.: Visibility histograms and visibility-driven transfer functions. *IEEE TVCG 17*, 2 (2011), 192–204. 2

[CSBI05] CAMARRI S., SALVETTI M.-V., BUFFONI M., IOLLO A.: Simulation of the three-dimensional flow around a square cylinder between parallel walls at moderate Reynolds numbers. In *XVII Congresso di Meccanica Teorica ed Applicata* (2005). 7

[CSC07] CORREA C., SILVER D., CHEN M.: Illustrative deformation for data exploration. *IEEE TVCG (Proc. Vis) 13*, 6 (2007), 1320–1327. 2

[CSFP12] CARNECKY R., SCHINDLER B., FUCHS R., PEIKERT R.: Multi-layer illustrative dense flow visualization. *CGF 31*, 3 (2012), 895–904. 2

[CWM*09] CHAN M.-Y., WU Y., MAK W.-H., CHEN W., QU H.: Perception-based transparency optimization for direct volume rendering. *IEEE TVCG (Vis) 15*, 6 (2009), 1283–1290. 2

[CWW13] CRANE K., WEISCHEDEL C., WARDETZKY M.: Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Trans. Graph. (Proc. SIGGRAPH) 32*, 5 (2013), 152:1–152:11. 3, 9

[DFRS03] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Suggestive contours for conveying shape. *ACM Trans. Graph. (Proc. SIGGRAPH) 22*, 3 (2003), 848–855. 2

[DWE02] DIEPSTRATEN J., WEISKOPF D., ERTL T.: Transparency in interactive technical illustrations. *CGF (Proc. Eurographics) 21*, 3 (2002), 317–325. 2

[ELC*12] EDMUNDS M., LARAMEE R. S., CHEN G., MAX N., ZHANG E., WARE C.: Surface-based flow visualization. *Computers & Graphics 36*, 8 (2012), 974–990. 2

[ELM*12] EDMUNDS M., LARAMEE R., MALKI R., MASTERS I., CROFT T., CHEN G., ZHANG E.: Automatic stream surface seeding: A feature centered approach. *CGF (Proc. EuroVis) 31*, 3 (2012), 1095–1104. 5

[FWT08] FREDERICH O., WASSEN E., THIELE F.: Prediction of the flow around a short wall-mounted cylinder using LES and DES. *JNAIAM 3*, 3-4 (2008), 231–247. 7

[GDN98] GRIEBEL M., DORNSEIFER T., NEUNHOEFFER T.: *Numerical Simulation in Fluid Dynamics, a Practical Introduction*. SIAM, 1998. 7

[GRT13] GÜNTHER T., RÖSSL C., THEISEL H.: Opacity optimization for 3D line fields. *ACM Trans. Graph. (Proc. SIGGRAPH) 32*, 4 (2013), 120:1–120:8. 2, 3, 4, 5, 6, 9

[GRT14] GÜNTHER T., RÖSSL C., THEISEL H.: Hierarchical opacity optimization for sets of 3D line fields. *CGF (Proc. EG) 33*, 2 (2014), to appear. 9

[HGH*10] HUMMEL M., GARTH C., HAMANN B., HAGEN H., JOY K. I.: IRIS: Illustrative rendering for integral surfaces. *IEEE TVCG (Proc. Vis) 16*, 6 (2010), 1319–1328. 2, 7, 8

[HL93] HOSCHEK J., LASSER D.: *Fundamentals of Computer Aided Geometric Design*. AK Peters, 1993. 4

[Hul92] HULTQUIST J. P. M.: Constructing stream surfaces in steady 3D vector fields. In *Proc. Vis* (1992), pp. 171–178. 5

[IFP96] INTERRANTE V., FUCHS H., PIZER S.: Illustrating transparent surfaces with curvature-directed strokes. In *Proc. Conference on Visualization* (1996), pp. 211–ff. 2

[JB10] JANSEN J., BAVOIL L.: Fourier opacity mapping. In *Proc. I3D* (2010), pp. 165–172. 6, 9

[JDA07] JUDD T., DURAND F., ADELSON E.: Apparent ridges for line drawing. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3 (2007). 2

[KKKK12] KUTZ B. M., KOWARSCH U., KESSLER M., KRÄMER E.: Numerical investigation of helicopter rotors in ground effect. In *AIAA Applied Aerodynamics* (2012). 7

[LCD06] LUFT T., COLDITZ C., DEUSSEN O.: Image enhancement by unsharp masking the depth buffer. *ACM Trans. Graph. (Proc. SIGGRAPH) 25*, 3 (2006), 1206–1213. 2

[Llo82] LLOYD S. P.: Least square quantization in PCM. *IEEE Information Theory 28*, 2 (1982), 129–137. 4

[MCTB11] MAULE M., COMBA J. L., TORCHELSEN R. P., BASTOS R.: A survey of raster-based transparency techniques. *Computers & Graphics 35*, 6 (2011), 1023–1034. 6

[MLP*10] MCLOUGHLIN T., LARAMEE R. S., PEIKERT R., POST F. H., CHEN M.: Over two decades of integration-based, geometric flow visualization. *CGF 29*, 6 (2010), 1807–1829. 2

[MSRT13a] MARTINEZ ESTURO J., SCHULZE M., RÖSSL C., THEISEL H.: Global selection of stream surfaces. *CGF (Proc. Eurographics) 32*, 2 (2013), 113–122. 5

[MSRT13b] MARTINEZ ESTURO J., SCHULZE M., RÖSSL C., THEISEL H.: Poisson-based tools for flow visualization. In *IEEE PacificVis* (2013), pp. 241–248. 5

[ND04] NIENHAUS M., DÖLLNER J.: Blueprints: Illustrating architecture and technical parts using hardware-accelerated non-photorealistic rendering. In *Proc. Graphics Interface* (2004), pp. 49–56. 2

[SLCZ09] SPENCER B., LARAMEE R. S., CHEN G., ZHANG E.: Evenly spaced streamlines for surfaces: An image-based approach. *CGF 28*, 6 (2009), 1618–1631. 2

[SMG*14] SCHULZE M., MARTINEZ ESTURO J., GÜNTHER T., RÖSSL C., SEIDEL H.-P., WEINKAUF T., THEISEL H.: Sets of globally optimal stream surfaces for flow visualization. *CGF (Proc. EuroVis) 33*, 3 (2014), to appear. 5

[ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3D shapes. *SIGGRAPH Comp. Graph. 24*, 4 (1990), 197–206. 2

[VG05] VIOLA I., GRÖLLER E.: Smart visibility in visualization. In *Proc. Computational Aesthetics* (2005), pp. 209–216. 2

[VKG04] VIOLA I., KANITSAR A., GRÖLLER E.: Importance-driven volume rendering. In *Proc. Vis* (2004), pp. 139–146. 2

[VPB*09] VERGNE R., PACANOWSKI R., BARLA P., GRANIER X., SCHLICK C.: Light warping for enhanced surface depiction. *ACM Trans. Graph. (Proc. SIGGRAPH) 28*, 3 (2009), 25:1–8. 2

[WGM*08] WANG L., GIESEN J., MCDONNELL K. T., ZOLLIKER P., MUELLER K.: Color design for illustrative visualization. *IEEE TVCG (Proc. InfoVis) 14*, 6 (2008), 1739–1754. 2

[YHGT10] YANG J. C., HENSLEY J., GRÜN H., THIBIEROZ N.: Real-time concurrent linked list construction on the GPU. *CGF (Proc. EGSR) 29*, 4 (2010), 1297–1304. 5, 6

[ZHXC09] ZHANG L., HE Y., XIE X., CHEN W.: Laplacian lines for real-time shape illustration. In *Proc. I3D* (2009), pp. 129–136. 2