Vision, Modeling, and Visualization (2024)
L. Linsen and J. Thies (Eds.)

Ray Tracing of Recirculation Surfaces

Daniel Stelter, Thomas Wilde and Holger Theisel

University of Magdeburg, Germany

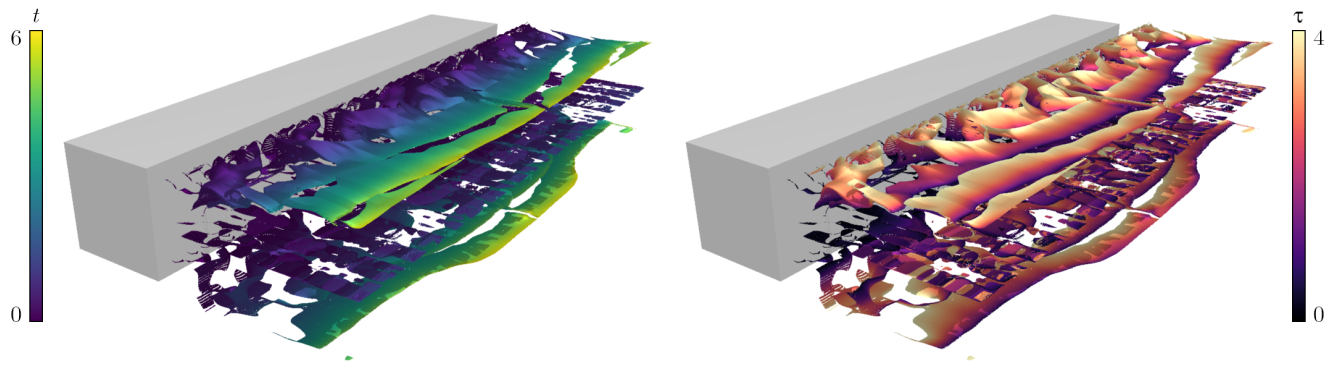


Figure 1: *Visualization of recirculation surfaces with our new ray tracing approach for the* SQUARE CYLINDER *dataset. Spatial positions are locations where particles advected by a flow return to this exact position. The color map in the left image indicates the start time t of a particle at this position. The color map in the right image depicts the advection time τ.*

Abstract

Recirculation in flows is an important phenomenon of dynamical systems as it is linked to numerous further properties and behaviors. A formal definition of recirculation surfaces has been introduced in previous work. However, the extraction and visualization of such surfaces is a highly complex challenge as they are 2-manifolds in the 5D space. Although an approach for the geometry extraction exists, there are still several unsolved problems, mainly connected to the computational effort and surface reconstruction. In this work, we propose a fundamentally different idea: Instead of extracting an explicit geometry, we apply a direct ray tracing approach. This way, we effectively circumvent the challenge of reconstructing the geometry. Additionally, we implement multiple strategies for an efficient computation. Due to this, we are able to provide a visualization of recirculation surfaces in a fraction of the computation time of existing approaches.

1. Introduction

Flow visualization is a subfield of computer visualization focused on graphically representing flow characteristics to enhance understanding. It serves as a tool for flow analysis, with example applications in vehicle design, meteorology, and medicine. Various features of flows can be visualized, with time-varying flows presenting a significant challenge.

This work focuses on the property of recirculation of particles in unsteady (i.e. time-dependent) flows. WILDE et al. [WRT18] provide a definition that can be described as follows: Considering a massless particle driven by a flow in a space, it is possible for the particle to return to exactly the same position where it started. This return to the starting position is termed *recirculation*. Investigating this property can offer deeper insights into the nature of the flow, and thus, giving more evidence for interpreting the dynamical behavior.

Finding such *recirculation points* in 3D flows spans a 5D search space with three spatial and two temporal dimensions: one for the start time and one for the advection time of a particle. WILDE et al. [WRT18] have found that such recirculation points form closed 2-manifolds in the 5D space. Additionally, they introduced a method to identify these recirculation surfaces. One part of their algorithm involves sampling the spatial dimensions with a multitude of axis-aligned lines. The result is a point cloud which consists of samples of the surface in 5D. Visualizing these identified points is highly complex. Although the authors were able to conceptually generate a surface reconstruction, they do not claim to achieve a sufficient sampling of the surfaces and left this as an open problem. Additionally, they point out extremely long runtimes as a disadvantage.

This work aims to describe and implement an alternative idea for the computation and visualization. Instead of constructing a point

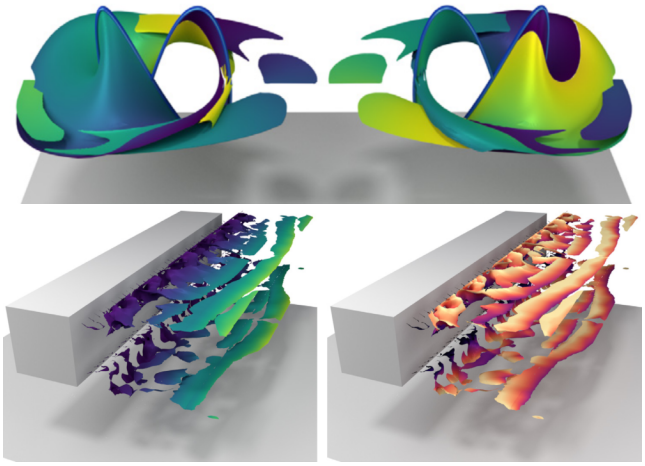


Figure 2: *Projections of recirculation surface extractions taken from* WILDE *et al. [WRT18] for the* 3D DOUBLE GYRE *(top) and the* SQUARE CYLINDER *(bottom) datasets. Color encodings in top and bottom left images refer to the start time t_0, and in bottom right image to advection time τ of a particle.*

cloud of samples, we apply a ray tracing approach. Surface intersections are found similarly as in the original work of WILDE et al. [WRT18], but are employed as rays of a camera. Thus, we also circumvent a possibly error-prone reconstruction entirely. Our focus lies on the direct creation of images rather than the complete determination and subsequent visualization of recirculation surfaces. This has shown to yield even richer visualizations compared to the current reconstruction strategy. Beyond that, our approach can provide results much faster as it does not rely on an extensive extraction in the whole domain.

The remaining work is structured as follows. Section 2 discusses related literature. Section 3 then describes our basic approach as well as multiple improvements of performance and visual quality. In Sections 4 and 5 we test our methods on multiple datasets, and subsequently discuss these results. Section 6 summarizes the findings of this work and provides an outlook.

2. Related Work and Background

Despite significant advancements over the last decades, flow visualization remains a highly active research area due to ongoing challenges and unsolved problems. Examples for recent works are visualizations of the topology of discontinuous vector fields [MDS23] and of uncertain stream lines [ZMRT23]. In this work, we are concerned with the recirculation of flows. We give a brief overview about related concepts and the foundations of this work.

Flow features and visualization. Flow visualization is concerned with making patterns and dynamics of fluid flows visible. Many different flow features have been investigated over the time. POST et al. [PVH*03] propose an overview about flow features and different techniques for their tracking and extraction. BUJACK et al. [BYH*20] present and discuss a multitude of methods for time-dependent topology. Lagrangian Coherent Structures, for instance, show the most attracting and repelling structures inside a dynamical system [SLM05; Hal15]. Vortices are an intuitively similar, but not identical concept to recirculation. These are regions for which the flow mainly rotates around a core line. GÜNTHER and THEISEL [GT18] provide a thorough overview. DENG et al. [DWL*19] apply a convolutional neural network for identifying vortex structures, achieving both high precision and efficiency. BIDDLE et al. [BKL20] examine the 4D structure of center vortices, revealing insights into their geometry and time-evolution.

Relevance of recirculation. There are different domains for which the property of recirculation plays an important role. For example, JACKSON et al. [JFL*18] observe the microphysical evolution of convective clouds and also investigate the influence of recirculation. SANTÍN et al. [SVPB22] present an approach to regulate internal recirculation in flows of biological wastewater treatments. OERTEL and SCHEMM [OS21] recently studied the advection of convective clouds and described that clouds move slower than their surroundings. They link this behavior to (re-)circulation processes inside the clouds and argue that this effect needs to be included in future simulation models.

Recirculation surfaces. Although recirculation is an important property of flows, it is challenging to formally describe this phenomenon. WILDE et al. [WRT18] propose a mathematical definition of recirculation as the loci where particles return to their original starting position after flow advection:

$$\overline{\overline{Y}} = \left\{ (\mathbf{x},t,\tau)^\top \in \mathbb{R}^3 \times \mathbb{R} \times \mathbb{R} : \mathbf{d}(\mathbf{x},t,\tau) = \mathbf{0} \wedge \tau \neq 0 \right\}$$

$$\mathbf{d}(\mathbf{x},t,\tau) = \frac{\phi(\mathbf{x},t,\tau) - \mathbf{x}}{\tau},$$

where the flow map ϕ returns the position of a particle starting at position $\mathbf{x}$ and time t after being advected by the time-dependent flow for τ time units. Thus, path lines (which are the trajectory of a particle) form a closed loop. Such loci form 2-manifolds in 5D, consisting of the 3D spatial position, the start time and the advection time. In order to extract recirculation points, they fix two spatial axes (e.g. $x = \hat{x}$ and $y = \hat{y}$) and construct the 3D vector field $\mathbf{d}_{\hat{x},\hat{y}}(z,t,\tau) = \mathbf{d}((\hat{x},\hat{y},z),t,\tau)$. Recirculation points are the zeros of $\mathbf{d}_{\hat{x},\hat{y}}$ which can be found using a standard approach for extracting isolated critical points from 3D steady grids [Wei08]. This requires a partition of the spatial lines into small segments to reliably perform the search. Via projection to the spatial 3D space, the surface is reconstructed using the *ball pivoting* algorithm [BMR*99] and visualized using color encodings for the start and advection time (see Figure 2). However, WILDE et al. point out challenges with both extreme runtimes and possibly insufficient surface reconstruction. HOFMANN and SADLO [HS19] applied the *dependent vectors operator* as an alternative approach to the problem, but their extractions exhibit holes and uneven surfaces.

3. Methods

This chapter presents methods for calculating recirculation surfaces using a ray tracer directly on given vector fields. Section 3.1 discusses the fundamental approach. Subsequently, various extensions to the algorithm are introduced. Sections 3.2 and 3.3 address the computation of lighting effects (shading and shadows) to render the surfaces with greater realism. In Section 3.4, a technique is proposed to achieve larger resolutions and anti-aliasing with reduced

computational effort. Furthermore, our methods partly rely on determining whether two recirculation points can be considered as neighbors of the 2-manifold in 5D. We propose our neighborhood estimation in Section 3.5.

3.1. Basic Ray Tracing Approach

Ray tracing builds on finding the closest intersection of any scene object along rays. However, computing intersections of a recirculation surface and visualizing it is not trivial. We follow the idea of WILDE et al. [WRT18] and extract the spatial 3D projection of the manifolds. Start and advection time are presented using two images with respective color encodings. For this problem we modify the technique which is reviewed in Section 2. Instead of fixing two spatial dimensions along axis-parallel lines, we restrict the search along the camera ray $\mathbf{r}(s) = \mathbf{v}_{eye} + s \cdot \mathbf{v}_{dir}$ with $s > 0$. Thus, we define our own 3D vector field $\mathbf{d}_\mathbf{r}(s,t,\tau) = \mathbf{d}(\mathbf{r}(s),t,\tau)$ which generalizes the search to arbitrary line segments. Similarly to WILDE et al., we can extract recirculation points on small line segments of a ray.

The next step is to choose which part of the ray has to be tested. Generally, one can only test the part of the ray which lies inside the flow domain. We define this interval as $[s_{\min}, s_{\max}]$. If a ray does not even intersect the domain, no test has to be performed at all. Beyond that, the recirculation surface is the most complex object in the scene. Thus, all other scene objects should be checked beforehand. If another object is intersected (at ray position s_{obj}), we differentiate three cases:

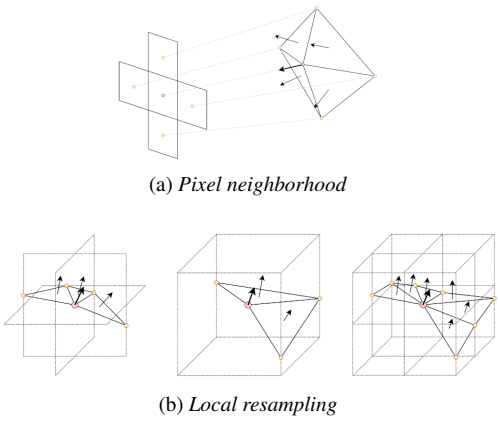
1. $s_{\max} < s_{\text{obj}}$ — test the whole interval
2. $s_{\min} < s_{\text{obj}} \leq s_{\max}$ — set $s_{\max}$ to s_{obj}
3. $s_{\text{obj}} \leq s_{\min}$ — skip entire test.

This interval has to be tested for recirculation points. As mentioned before, reliably extracting recirculation points requires to split the ray into multiple short line segments. For each of them we can perform the extraction method of WILDE et al. As one is only interested in the closest intersection to the ray origin, we iterate starting from position $s_{\min}$. The routine ends after the first occurrence of a recirculation point, or if the whole interval was tested without an intersection. For the visualization we follow the example of WILDE et al. and create two images. Both show the same 3D projection of the recirculation surface, but with two color encodings: one for the start time t and one for the advection time τ of the particle.

3.2. Shading

After introducing the basic ray tracing principles, one can now render basic visualizations of recirculation surfaces. To achieve better spatial impressions, we apply Phong shading [Pho75]. However, this requires a 3D surface normal which is not readily available. We propose two methods to estimate normals of recirculation surfaces.

Pixel neighborhood. The strategy estimates normals directly from the sampling of the initial image. We consider a pixel as well as all its neighboring pixels and assume the following: First, for each of these pixels the ray tracer found a recirculation point. Second, all of these points can be considered neighbors on the recirculation surface in 5D. If these statements hold, one can trivially estimate the spatial 3D normal by constructing four triangles as shown in Figure 3a. The mean of their normals is our estimate.



(a) *Pixel neighborhood*

(b) *Local resampling*

Figure 3: *Strategies for computing normals of recirculation surfaces. The pixel neighborhood strategy (a) reuses neighboring intersections found with camera rays. For local resampling (b), new line segments are sampled with three layouts, from left to right: cross, cube, and combined. Respectively, the intersections are triangulated and the average of all normals is returned.*

However, this poses more problems. The decision whether two recirculation points are neighbors in the 5D space is not trivial. A fixed threshold for the 5D Euclidean distance is insufficient, as the minimal possible spatial distances also depend on the resolution of the image. Additionally, in case of ray tracing with a perspective camera, the distance from the viewpoint is a second factor. A solution to this problem is discussed in Section 3.5. Beyond that, there are cases where not all neighboring pixels refer to 5D neighbors, or no recirculation point exists at all. It suffices to construct at least one valid triangle. Otherwise, this routine is unable to return a normal.

Local resampling. Another approach involves conducting an entirely new search for neighbors near the 5D point of interest $\hat{\mathbf{x}} = (\mathbf{x}, t, \tau)$. For this purpose, we seed multiple line segments around $\mathbf{x}$ and search for further recirculation points. Figure 3b illustrates two sampling setups, as well as their combination. Start and end points of the line segments are defined using a small offset d. The corners of the setups are:

- *Cross:* $\{\mathbf{x} + (\pm d, \pm d, 0), \mathbf{x} + (\pm d, 0, \pm d), \mathbf{x} + (0, \pm d, \pm d)\}$
- *Cube:* $\{\mathbf{x} + (\pm d, \pm d, \pm d)\}$

This search results is a set of recirculation points which also has to be filtered by 5D neighborhood to $\hat{\mathbf{x}}$. Afterwards, one can apply a standard triangulation algorithm and use the average of all normals for the shading.

In this work, the setups are applied as follows. Estimating normals with pixel neighbors is very cheap, and thus, the first choice. If no normal can be constructed, we search for normals by locally sampling new lines. First, neighboring recirculation points are searched with the *cross* setup. In case that no triangles can be constructed, we additionally search intersections with the *cube* setup, resulting in their combination. If this still yields no triangles, d is cut in half and the search is repeated. In very rare cases, even this search might not find a normal. We handle this by not applying any shading.

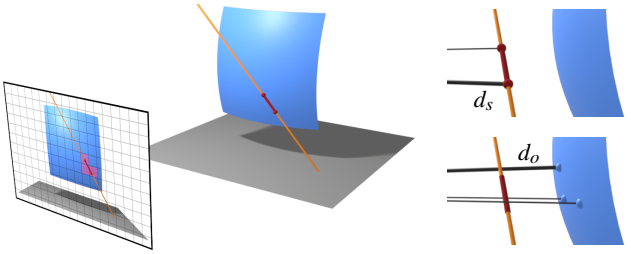


Figure 4: *Decision process for efficient shadow computations. Consider the orange shadow ray with a red line segment $\mathcal{S}$. (a) shows the image plane from the camera perspective. The projection of the red segment results in three intersected pixels which are marked red. (b) presents the scene from a side view. Black lines depict the distance vectors of the ray segment (top) and the object intersections of the three marked pixels (bottom) to the camera origin.*

3.3. Efficient Shadow Tests

Next, we investigate the principle of shadows. This requires an additional surface intersection test in direction of the light source. The naive approach would be to apply the exact same routine as in Section 3.1. We propose a more efficient method which reuses the original sampling of the image. From the perspective of the camera, it is known where recirculation points have already been found. This is valuable information since it gives evidence that there are no intersections along the entire ray up to the first occurrence. We prevent the test of line segments which lie in front of these intersections.

Given is the shadow ray $\mathbf{s}(s)$ and a line segment $\mathcal{S}$ on this ray. The decision whether the segment should be tested for intersections is based on a distance comparison to the camera. We determine two reference distances: d_s for the segment $\mathcal{S}$, and d_o for object intersections of nearby camera rays. A simple and careful choice for d_s is to simply use the point of $\mathcal{S}$ with the largest distance to the camera, i.e. , either its start or end point. For d_o, we project $\mathcal{S}$ onto the image plane. The set of pixels which are intersected by the projected line segment yield the nearby camera rays. For each of these rays, we obtain the distance to the closest intersection in the scene. We define d_o as the minimum of these distances. If $d_s < d_o$ holds or if there are no intersections at all, the ray segment is assumed to be in front of all present objects, and the intersection test is skipped. Figure 4 illustrates how to obtain all relevant rays.

Although the described method gives evidence that $\mathcal{S}$ does not contain a surface intersection, it is not a proof. Consider a case where a part of a surface is nearly parallel to the camera rays. As the estimation relies on the image sampling, the value of d_o might be too large. The only way to prevent this is a test of the entire shadow ray. For this reason, we apply a post-processing routine after the initial shadow computation. This is applied to each pixel that has a 5D neighbor which is in shadow. Note that the initial shadow test already checked a subset of all ray segments. Thus, only the initially skipped segments have to be tested. We apply this routine recursively until no new shadows are found.

3.4. Increasing Resolution & Anti-Aliasing

In this section, we present a strategy to reduce the runtime for large image resolutions, as well as for anti-aliasing. Both are able to significantly improve the visual quality and perception. In the following, we distinguish between *initial* and *new* images, pixels, and camera rays. As for shadow computations (see Section 3.3), we reuse the information of an initial image. Under the assumption of a sufficiently dense initial image sampling, we expect surface intersections for nearby rays at roughly the same depth. Therefore, our idea is to prevent intersection tests for new rays if no nearby initial ray found an intersection either.

We start by explaining the ideas for increasing the initial image resolution $U_{\text{init}} \times V_{\text{init}}$ by an upscale factor $f > 1$. Thus, the new image resolution is defined as $(f \cdot U_{\text{init}}) \times (f \cdot V_{\text{init}})$. Given is a new camera ray with pixel coordinates $(u_{\text{new}}, v_{\text{new}})$. One can easily identify the corresponding initial pixel (i.e. , the one which „contains" the new ray) and nearby neighboring initial rays. If the initial ray and the new ray are identical, the result can even be reused. Else, if none of these neighboring rays found an intersection, we assume that the ray $(u_{\text{new}}, v_{\text{new}})$ does not intersect the surface, too. Therefore, it is not checked at all. Else, we consider the intersections of the initial rays. The minimum distance to the camera origin is used as the start for the search of the new ray. For a more reliable application, we also apply a small offset in the direction of the camera. This can drastically reduce the number of intersection test, and thus, the runtime. Exactly the same idea can be applied for anti-aliasing. We apply a Quincunx sampling, where each initial pixel is sampled at its four corners. Consequently, the pixel is colored by averaging the colors of these four corner samples and the initial central sample.

In most cases, this principle works well. However, there are two problems. First, there can be structures which were not sampled by the initial image due to too low resolutions. Thus, it will also not be detected by the new sampling. This can only be avoided by larger initial image resolutions. Second, in rare cases the estimated start position can be behind the actual recirculation surface. For instance, there can be surface structures which are nearly parallel to the camera rays. Therefore, the structure might not be fully sampled. Again, we apply a post-processing step which tests the previously omitted ray segments. This requires a detection of all rays with possibly missing intersections. The set consists of all new rays for which at least one of these statements hold:

1. the ray found no intersection, but at least one neighbor did
2. the ray found an intersection, but at least one neighbor which is *no 5D neighbor* lies in front

Then, the part of the ray up to the original start position also has to be tested. This is applied recursively until no new neighbors are found. However, it is still unclear how to determine whether two recirculation points are considered as 5D neighbors. This is solved in the next section.

3.5. Neighborhood Estimation in 5D

In Sections 3.2 and 3.4 we introduced methods that rely on the estimation of neighborhoods in 5D. As explained previously, a Euclidean distance threshold for the 5D points is insufficient. First,

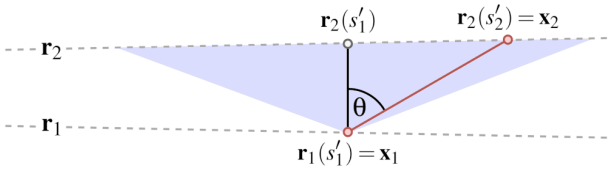


Figure 5: *Spatial test of two recirculation points on nearby rays* $\mathbf{r}_1$ *and* $\mathbf{r}_2$*. The blue background illustrates the maximum allowed angle* $\theta_{\max}$*. Thus, this example pictures a successful test.*

the relation between the spatial and both temporal dimensions may be arbitrary. Second, even for the rays in 3D space, the minimal possible distance depends on the image resolution and the depth of the intersections.

For these reasons we decouple the spatial and temporal dimensions. Given are the neighboring rays $\mathbf{r}_1(s_1)$ and $\mathbf{r}_2(s_2)$ with recirculation points $\hat{\mathbf{x}}_1 = (\mathbf{x}_1, t_1, \tau_1)$ and $\hat{\mathbf{x}}_2 = (\mathbf{x}_2, t_2, \tau_2)$ such that $\mathbf{r}_1(s_1') = \mathbf{x}_1$ and $\mathbf{r}_2(s_2') = \mathbf{x}_2$. The spatial neighborhood criterion is illustrated in Figure 5. It requires the angle θ between $(\mathbf{r}_2(s_1') - \mathbf{x}_1)$ and $(\mathbf{x}_2 - \mathbf{x}_1)$ which is compared to a threshold $\theta_{\max}$. The rationale is that the best-case scenario is both intersections appearing at the exact same depth. Using an angle to this optimal difference also handles the problem of diverging rays at larger depths. For the start and advection times we introduce maximum thresholds $\tilde{t}_{\max}$ and $\tilde{\tau}_{\max}$ relative to the spatial distance. Thus, the final test is the following:

$$(\theta \leq \theta_{\max}) \wedge \left(\frac{|t_2 - t_1|}{\|\mathbf{x}_2 - \mathbf{x}_1\|} \leq \tilde{t}_{\max} \right) \wedge \left(\frac{|\tau_2 - \tau_1|}{\|\mathbf{x}_2 - \mathbf{x}_1\|} \leq \tilde{\tau}_{\max} \right)$$

This is a simple test which can be performed in real-time as soon as the intersections have been computed. Due to this, the three threshold parameters can also be chosen interactively by visualizing their influence to the user. For each pixel, it can be determined which neighbors can be used for normal computations (Section 3.2) and which rays must be post-processed after increasing the resolution (Section 3.4).

4. Results

We tested our ray tracing approach on two unsteady flow datasets. The respective setups for all experiments can be taken from Table 4. Since the approach of WILDE et al. [WRT18] is the only other method for recirculation surface extraction, we compare to their results for two datasets. This comparison includes visualization and extraction quality (see Figures 2 and 9), as well as runtimes. Regarding the runtimes, we executed a reference implementation provided by the original authors [Wil22]. It only includes the initial sampling of recirculation points and does neither apply local refinement nor reconstruction. We used the same domains as in their work, i.e. , the visualizations in Figure 2 match the values in Table 4. The experiments were performed on a system using two AMD Epyc 7543 CPUs (2.8 GHz, 32 cores each) and both algorithms took advantage of parallelization.

4.1. 3D Double Gyre

This dataset is an expansion of the well-known original DOUBLE GYRE of SHADDEN et al. [SLM05]. WILDE et al. added a third spatial component which exhibits multiple recirculation surfaces.

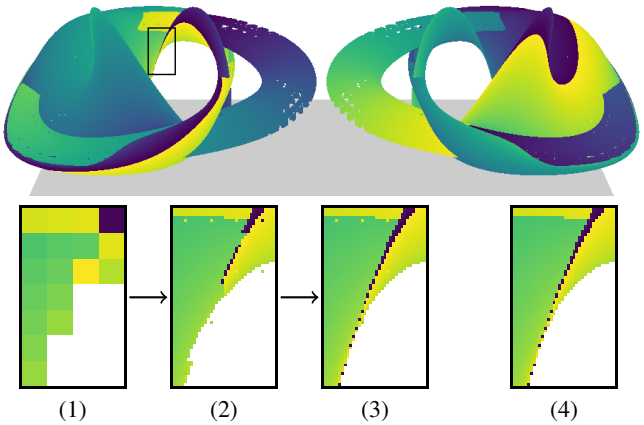


Figure 6: *Pipeline for the resolution increment for the* 3D DOUBLE GYRE*. It shows the process from a low base resolution (1) to a larger resolution (2) with post-processing (3). For a comparison, (4) shows the result of a direct sampling. The results are nearly identical.*

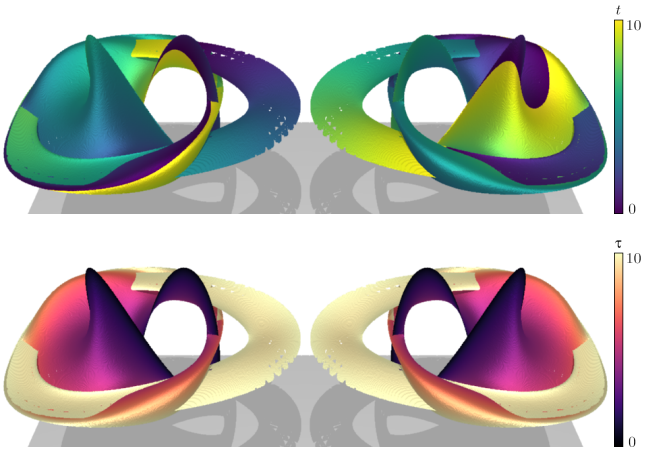


Figure 7: *Final results with our ray tracing approach for the* 3D DOUBLE GYRE *dataset. This includes shading, shadows, and anti-aliasing on a resolution of* 900×270 *pixels.*

Although it is a relatively simple dataset, it already highlights the challenges of self-intersections of the projected recirculation surfaces. In their work, WILDE et al. used a grid resolution of $200 \times 100 \times 100 \times 50 \times 50$ for the three spatial and two temporal dimensions. This coincides to our segment lengths in Table 4.

For this dataset, we consider two image resolutions. The first resolution (100×30 pixels) was used as a base image for the refinement (Section 3.4). Its computation took less than 3 minutes. We increased the image resolution with $f = 9$, meaning that the second resolution is 900×270. Increasing the resolution needed 26 minutes, with 10 more minutes for post-processing along edges. In comparison, we also computed the large image resolution directly (i.e. without the methods from Section 3.4) which took 216 minutes. Figure 6 shows this with a closeup of a region with a nearly parallel

Dataset	Domain			Segment lengths			Neighborhood parameters		
	$\mathbf{x}$	t	τ	$\mathbf{x}$	t	τ	θ_{max}	$\tilde{t}_{max}$	$\tilde{\tau}_{max}$
3D DOUBLE GYRE	$[0,2]\times[0,1]\times[0,1]$	$[0,10]$	$[0,10]$	0.01	0.2	0.2	85°	60	60
SQUARE CYLINDER	$[0.5,2.5]\times[-0.65,0.65]\times[0,6]$	$[0,6]$	$[0,4]$	0.005	0.2	0.2	85°	20	20

Table 1: *Setup parameters for our experiments. Segment lengths of* $\mathbf{x}$ *refer to the line segments of our rays. For* t *and* τ*, these are lengths of cell edges for the extraction of critical points (see [Wei08]). Same parameters are also applied to the method of* WILDE *et al. [WRT18].*

surface to the camera rays. Even in this case, the post-processing was able to achieve nearly identical results as for the basic sampling. Indeed, it even extracted 82 more recirculation points than the basic sampling. Additionally, it was far more efficient: Including the low resolution computation, the runtime was just 39 minutes. The reference implementation of WILDE et al., on the other hand, took about 79 minutes. We emphasize that this does not include local refinement. In their work, this took an additional 36% of the time of the initial sampling.

Next, we additionally applied shading, shadows, and anti-aliasing. All of these are optional, but can improve the perception of the results. Computing the normals for the shading was relatively inexpensive, taking around 1 minute. In contrast, Shadows needed 34 minutes (28 for initial tests, 6 for post-processing). This mainly comes due to the bottom plane which contains most of the shaded points. The final result can be seen in Figure 7. For anti-aliasing, initial computation and refinement had a runtime of 10 minutes each.

As compared in Figure 9, we can visualize many more recirculation points than WILDE et al. Especially the orbit around the gyres and for regions with self-intersections show much richer surfaces.

4.2. Square Cylinder

This dataset is the result of a direct numerical Navier Stokes simulation proposed by CAMARRI et al. [CIBS06]. It simulates an incompressible flow in a system with solid walls and a square cylinder which is placed symmetrically between two walls. In this work we use a uniformly resampled version of Tino Weinkauf which was also used in [VWTS08]. Again, the segment lengths in Table 4 coincide to the resolutions of WILDE et al.

As for the first dataset, we started with a low resolution of 200×120 pixels. The basic sampling took around 3 minutes. We increased the resolution to 200×120 by applying the method from Section 3.4. This required 32 minutes, with an additional 18 minutes for post-processing. In contrast, a direct sampling of the large resolution needed 78 minutes. However, this dataset contains few very fine recirculation surfaces. Applying the resolution increment missed out the ones which were not captured by the low resolution sampling (see Figure 8). In comparison, the reference implementation of WILDE et al. had a runtime of over 8 hours. In their work, local refinement took an additional 58%. This comes mainly due to their very dense sampling grid.

For the visualization in Figure 1 we also applied further advancements. Computing the samples for anti-aliasing needed 24 minutes with 36 additional minutes for post-processing. For shading, over 90% of all surface normals were computed with the *pixel neighborhood* method in less than a second. The computation of the

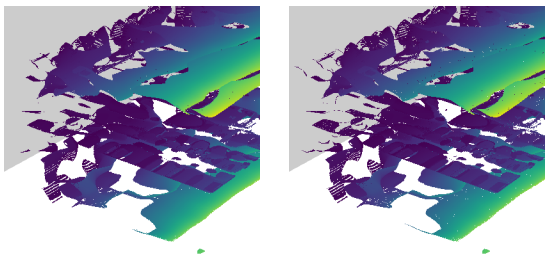


Figure 8: *Closeup comparison for the* SQUARE CYLINDER *dataset for the post-processed resolution increment (left) and the direct sampling of the large resolution (right). Some of the very fine structures are not captured by the increment strategy, but the vast majority has similar quality.*

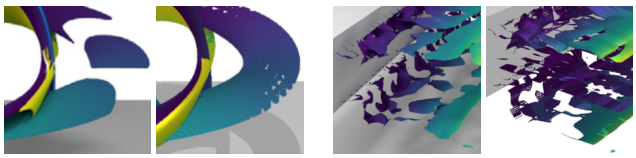


Figure 9: *Direct visualization comparison of extraction techniques for the* 3D DOUBLE GYRE *(left) and the* SQUARE CYLINDER *datasets. Respectively, the left image is the result of* WILDE *et al., the right image comes from our ray tracer. In both examples, we achieve much richer visualizations.*

remaining 10% with *local resampling* took 148 minutes. However, computing the other 90% with *local resampling* only needed 17 minutes in total. Thus, we expect the extreme runtimes to stem from an implementation mistake. Fixing it should drastically reduce this runtime. Shadows took another 24 minutes which already includes post-processing.

Similarly to the 3D DOUBLE GYRE the comparison in Figure 9 shows a significant difference in terms of extracted recirculation surfaces. We visualize structures at mid-height which do not appear at all for WILDE et al.

5. Discussion

5.1. Visualization Quality

The results of WILDE et al. and our methods differ in one important way: They sample and reconstruct the recirculation surfaces directly. We, on the other hand, apply a direct visualization using ray tracing. A significant advantage of our direct approach is that all intersections with the recirculation surfaces can be represented as they were actually found. It is notable that our results showed

far richer extractions (see Figure 9). In case of the 3D DOUBLE GYRE, there are two major differences. First, the outer orbits are closed which is not the case for WILDE et al., and second, in regions of self-intersections they strongly struggle to reconstruct the surfaces which was no problem for our ray tracer. This was even more significant for the SQUARE CYLINDER where we found many more surfaces at medium height of the cylinder. These structures show an extreme amount of self-intersections and do not appear at all in Figure 2. We assume that the surface reconstruction for the corresponding areas failed, preventing them from being represented. This underscores the advantage of our direct visualization.

For our visualizations, the exact structure of the surface is irrelevant, as the detected points are directly mapped. This also means that all recirculation points can be visualized as-is without any intermediate steps. However, at some points, tiny gaps appeared in the surfaces. For the 3D DOUBLE GYRE, this was limited to the orbiting surface where the advection time was near to the maximum of the searched domain. Increasing the domain for τ showed to expand the orbits and to close these gaps. Therefore, this is not a weakness of our method, but comes due to the specified search space. In case of the SQUARE CYLINDER, on the other hand, few surfaces showed thin holes although both t and τ values are not near to their limits. This gives evidence that the search parameters might be slightly too low for these rays. Increasing the accuracy should improve the quality, but also increase the runtime.

Further aspects of the quality include the shading, shadows and anti-aliasing. We claim to achieve well perceivable visualizations. However, one obvious limitation in our results is that we only considered hard shadows. Especially for the 3D DOUBLE GYRE, the soft shadows applied by WILDE et al. give a more pleasing effect. In theory, one could also expand our methods to soft shadows, but we expect unreasonably long runtimes.

5.2. Selection of a Camera Pose

One of the most important decisions for rendering a scene is the choice of the camera settings. This determines whether and how individual objects are visible and what information the viewer receives. In a scene with common geometric objects, the camera pose can easily be selected interactively. However, for an unknown dataset it is unclear where (and even if) there is a recirculation surface. Its rendering is very complex and time-consuming after all which prevents such an interactive search. Thus, a good practice is to start with a camera pose which contains the whole domain and render the scene at a low resolution. This allows a basic understanding of the initially unknown surface. After determining a suiting pose, the initial rendering may be reused for the resolution increment process described in Section 3.4. Anyway, we acknowledge that WILDE et al. do not face these problems once they reconstructed the surfaces. Additionally, rendering the scene with different poses is comparatively easy for them. In contrast, our approach requires a complete recalculation.

5.3. Runtime

Visualizing recirculation surfaces in general is a very complex and time-consuming process. We proposed multiple effective methods to save some expensive computations. This includes an efficient normal computation for shading, faster shadow ray computations, and increasing image resolutions as well as efficient anti-aliasing depending on an initial image sampling. The primary goal of these methods is to reduce runtimes while preserving the quality of the results. While indeed providing high quality results in this work, our computations still resulted in runtimes of up to a few hours. However, we argue that even longer runtimes as for WILDE et al. are acceptable since our extractions are richer, and thus, better. Beyond that, the runtimes can easily be influenced by lowering the quality. Even lower image resolutions than presented in this work may give sufficient results. This also decreases all further computations, including shading, shadows, and anti-aliasing. Additionally, these features are even optional. Thus, if one is not interested in a high-quality visualization, they can be left out completely. This control of quality versus runtime is much easier for us than for WILDE et al. We further discuss this in Section 5.4.

An additional application for the efficient resolution increase is progressive data analysis [FP16]. In a scenario where one needs the results as fast as possible, our ray tracer can progressively improve the image resolution. As shown, images with low resolutions can be created within few minutes. For example, new pixels can be added in between existing ones. Thus, the quality can be improved step by step.

5.4. System Parameters

Determining recirculating surfaces is a highly complex problem that requires a multitude of algorithms, all coming with their own parameters. WILDE et al. have already discussed the parameters used in their work. Amongst others, this includes the choice of an ODE solver and a critical point extractor with respective parameters. We restrict to a comparison of differences to our methods and refer to their work for a more in-depth discussion.

To reconstruct recirculation surfaces using their extracted samples, they require a heuristic for the neighborhoods of critical points at different time steps. In this work, another heuristic for neighborhood estimation was introduced which is tailored to the ray tracing task. We apply it for the estimation of 3D surface normals and for post-processing along surface edges. It requires three threshold values $\theta_{\max}$, $\tilde{\tau}_{\max}$ and $\tilde{\theta}_{\max}$ that can be interactively chosen. Additionally, these parameters have no side effects with other search parameters. This makes their selection relatively easy. As both works require parameters, the complexity does not increase for our method.

For WILDE et al., one must choose the resolution of the initial grid for sampling recirculation points. This, however, is under the risk of too sparsely selected grid resolution which could lead to bad reconstructions. Additionally, this might not be noted until performing the whole process up to the visualization. In that case, the entire expensive computation must be discarded and redone with a finer grid. Too dense samplings, on the other hand, lead to very long runtimes as it grows exponentially with increasing grid resolutions. For the ray tracing application, the spatial grid was replaced by two factors: length per line segment, and image resolution. As explained in Section 5.2, it is advised to start with low image resolutions. This also helps to recognize problems caused by the search parameters early on.

5.5. Limitations & Future Work

In the last sections, we already discussed advantages, but also limitations of our methods. The most prominent limitation is the inflexibility of the camera pose, i.e. , computing images from different angles requires a complete recomputation. Similarly, one has to search for a suiting setup before generating high-quality results. Another limitation of the efficient resolution increase is that it entirely omits fine structures if the initial image did not find it either.

We see many options for future work. Especially, there is potential for further improvements of our current methods. One example could be to transfer the idea of efficient shadow computations to new camera poses. A similar optimization could be done for slight shifts of the camera, i.e. , if one would like to rotate or move the view. These extensions would tackle the limitation of inflexible camera poses. Another goal might be to further improve the efficiency. One idea is to apply adaptive ray line segments. For example, in regions where all path lines for all start and advection time combinations diverge, the segments might be larger. However, this is not trivial and might be a future optimization. Additionally, our shadow computations are still comparatively expensive. There might be potential to find a similar application of the resolution increase, i.e. , using the assumption that nearby pixels may have shadow rays with nearby surface intersections.

6. Conclusions

In this work we presented a new ray tracing approach for visually extracting recirculation surfaces. Such surfaces consist of points for which material which is transported by a flow returns to its initial starting position. Our new method is an alternative to the only other existing approach proposed by WILDE et al. [WRT18]. Instead of applying a complex and error-prone surface reconstruction, our direct visualization effectively circumvents this problem. Therefore, our extraction presents far more surface structures than the work of WILDE et al. Additionally, we contributed methods to efficiently increase image resolutions and apply shading, shadows, and antialiasing effects. Runtimes depend strongly on the requirements of the resulting images, i.e. , the final resolution or whether effects like shadows should be applied, but can be much shorter than the extraction of WILDE et al. We see a variety of options to further optimize and advance our current methods. This could reduce runtimes even further and make our approach more flexible.

References

[BKL20] BIDDLE, JAMES C, KAMLEH, WASEEM, and LEINWEBER, DEREK B. "Visualization of center vortex structure". *Physical Review D* 102.3 (2020), 034504 2.

[BMR*99] BERNARDINI, FAUSTO, MITTLEMAN, JOSHUA, RUSHMEIER, HOLLY, et al. "The ball-pivoting algorithm for surface reconstruction". *IEEE transactions on visualization and computer graphics* 5.4 (1999), 349–359 2.

[BYH*20] BUJACK, ROXANA, YAN, LIN, HOTZ, INGRID, et al. "State of the art in time-dependent flow topology: Interpreting physical meaningfulness through mathematical properties". *Computer Graphics Forum*. Vol. 39. 3. Wiley Online Library. 2020, 811–835 2.

[CIBS06] CAMARRI, SIMONE, IOLLO, ANGELO, BUFFONI, MARCELO, and SALVETTI, MARIA VITTORIA. "Simulation of the three-dimensional flow around a square cylinder between parallel walls at moderate Reynolds numbers". *XVII Congresso di Meccanica Teorica ed Applicata*. 2006, 1000–1012 6.

[DWL*19] DENG, LIANG, WANG, YUEQING, LIU, YANG, et al. "A CNN-based vortex identification method". *Journal of Visualization* 22 (2019), 65–78 2.

[FP16] FEKETE, JEAN-DANIEL and PRIMET, ROMAIN. "Progressive analytics: A computation paradigm for exploratory data analysis". *arXiv preprint arXiv:1607.05162* (2016) 7.

[GT18] GÜNTHER, TOBIAS and THEISEL, HOLGER. "The state of the art in vortex extraction". *Computer Graphics Forum*. Vol. 37. 6. Wiley Online Library. 2018, 149–173 2.

[Hal15] HALLER, GEORGE. "Lagrangian coherent structures". *Annual review of fluid mechanics* 47 (2015), 137–162 2.

[HS19] HOFMANN, LUTZ and SADLO, FILIP. "The dependent vectors operator". *Computer Graphics Forum*. Vol. 38. 3. Wiley Online Library. 2019, 261–272 2.

[JFL*18] JACKSON, ROBERT, FRENCH, JEFFREY R, LEON, DAVID C, et al. "Observations of the microphysical evolution of convective clouds in the southwest of the United Kingdom". *Atmospheric Chemistry and Physics* 18.20 (2018), 15329–15344 2.

[MDS23] MIFTARI, EGZON, DURSTEWITZ, DANIEL, and SADLO, FILIP. "Visualization of Discontinuous Vector Field Topology". *IEEE Transactions on Visualization and Computer Graphics* (2023) 2.

[OS21] OERTEL, ANNIKA and SCHEMM, SEBASTIAN. "Quantifying the circulation induced by convective clouds in kilometer-scale simulations". *Quarterly Journal of the Royal Meteorological Society* 147.736 (2021), 1752–1766 2.

[Pho75] PHONG, BUI-TUONG. "Illumination for computer generated pictures". *Communications of the ACM* 18.6 (1975), 311–317 3.

[PVH*03] POST, FRITS H, VROLIJK, BENJAMIN, HAUSER, HELWIG, et al. "The state of the art in flow visualisation: Feature extraction and tracking". *Computer Graphics Forum*. Vol. 22. 4. Wiley Online Library. 2003, 775–792 2.

[SLM05] SHADDEN, SHAWN C, LEKIEN, FRANCOIS, and MARSDEN, JERROLD E. "Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows". *Physica D: Nonlinear Phenomena* 212.3-4 (2005), 271–304 2, 5.

[SVPB22] SANTÍN, I, VILANOVA, R, PEDRET, C, and BARBU, MARIAN. "New approach for regulation of the internal recirculation flow rate by fuzzy logic in biological wastewater treatments". *ISA transactions* 120 (2022), 167–189 2.

[VWTS08] VON FUNCK, WOLFRAM, WEINKAUF, TINO, THEISEL, HOLGER, and SEIDEL, HANS-PETER. "Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments". *IEEE Transactions on Visualization and Computer Graphics* 14.6 (2008), 1396–1403 6.

[Wei08] WEINKAUF, TINO. "Extraction of topological structures in 2D and 3D vector fields". PhD thesis. Magdeburg, Univ., Diss., 2008, 2008 2, 6.

[Wil22] WILDE, THOMAS. *RecirculationSurfaces*. https://github.com/Thomas-Wilde/RecirculationSurfaces. 2022 5.

[WRT18] WILDE, THOMAS, RÖSSL, CHRISTIAN, and THEISEL, HOLGER. "Recirculation surfaces for flow visualization". *IEEE transactions on visualization and computer graphics* 25.1 (2018), 946–955 1–3, 5–8.

[ZMRT23] ZIMMERMANN, JANOS, MOTEJAT, MICHAEL, RÖSSL, CHRISTIAN, and THEISEL, HOLGER. "Uncertain Stream Lines". (2023) 2.