Generation of Adaptive Streak Surfaces Using Moving Least Squares

Harald Obermaier[1,2], Martin Hering-Bertram[2], Jörg Kuhnert[2], and Hans Hagen[1]

1 University of Kaiserslautern, Germany
2 Fraunhofer ITWM Kaiserslautern, Germany

Abstract

We introduce a novel method for the generation of fully adaptive streak surfaces in time-varying flow fields based on particle advection and adaptive mesh refinement. Moving least squares approximation plays an important role in multiple stages of the proposed algorithm, which adaptively refines the surface based on curvature approximation and circumradius properties of the underlying Delaunay mesh. We utilize the grid-less Moving Least Squares approximation method for both curvature and surface estimation as well as vector field evaluation during particle advection. Delaunay properties of the surface triangulation are guaranteed by edge flipping operations on the progressive surface mesh. The results of this work illustrate the benefit of adaptivity techniques to streak surface generation and provide the means for a qualitative analysis of the presented approach.

1998 ACM Subject Classification I.6.6 Simulation Output Analysis

Keywords and phrases scattered, flow, streak surface, adaptivity, moving least squares

Digital Object Identifier 10.4230/DFU.Vol2.SciViz.2011.260

1 Introduction

Providing data for the evaluation of aerospace prototypes, industrial mixing processes, and many other applications of flowing liquids or gases is the result of more and more accurate simulations in *Computational Fluid Dynamics* (CFD). Analysis of the vector fields of these flow simulations is heavily dependent on the range of available visualization methods, as the information gathered from standard methods like direct volume rendering is limited and of neglectable expressive power when applied globally. This emphasizes the need for sophisticated feature-based visualization techniques, which has been the topic of active research for a number of years, leading to the definition of integral curves and surfaces. Generation and rendering of these surfaces in simulated flow fields is a well-established technique in the field of vector field visualization, whose homogeneous visual properties allow an in-depth analysis of the behavior of connected components of flow fields. While efficient methods to adaptively generate such surfaces in stationary vector fields and some generalizations like adaptive time surfaces are state-of-the-art, there are no known methods for fully adaptive streak surface generation as presented in this work. An important application of surface feature extraction is the visualization of separation topology in three-dimensional data sets what, due to the absence of adaptive streak surfaces, has been limited to the stationary vector field case so far. In comparison to surface definitions where merely the trace of a specific single particle or curve is tracked, as is the case in path surfaces, the definition of streak surfaces is capable of visualizing phenomena such as smoke and dye-advection, efficiently showing the movements of distinct, continuously seeded regions over time. In

practice, they may be used to visualize material boundaries in mixing processes or, more general, a time-varying analogon of the stationary three-dimensional separatrix definition, leading to a non-stationary form of three-dimensional vector field topology. Therefore, streak surfaces form the basis for topological volume segmentation in time-dependent vector fields. The major challenge of adaptive streak surface construction is the high complexity of mesh refinement arising from the new time dimension of the well-known advancing front definition introduced by Hultquist [11]. In every time step of a time-varying velocity field, not only a front of a few stream lines or particle traces needs to be examined, but the whole surface has to be analyzed with respect to adaptivity measures. In this work, we solve the problem of adaptive streak surface integration with the help of *Moving Least Squares* (MLS) approximation for particle advection as well as surface estimation and refinement based on Delaunay meshes. MLS facilitates accurate surface integration independent of any underlying computational meshes that may be created by the specific tool used for CFD simulation. In fact, the presented test data sets are mesh-less and obtained from a grid-less *Finite Pointset Method* (FPM) [18], whose interpolant is also based on MLS. Delaunay meshing helps to approximate surface particle densities and provides a basic triangulation for visualization. The challenge of time coherent surface generation and rendering is overcome by surface particle integration, adaptive particle tracing, and look back methods.
The work presented in this paper has the following main contributions to the community of vector field visualization:

- Fully adaptive grid-less streak surface generation
- Introduction of MLS into the generation of time-varying integral surfaces
- Integral surface refinement by time-dependent Delaunay based particle density adaptation

In section 2 we give an overview of the related work that has been published in the field of surface generation and Delaunay meshing. We present the concept of streak surfaces in section 3. The mathematical definition of the Moving Least Squares approximation method is given in section 4. Section 5 describes the steps of our algorithm for adaptive streak surface generation in detail. Numerical examples are provided in section 6. Section 7 concludes this paper and gives an outlook on future work.

2 Related Work

Adaptive (stream-) surface generation in stationary flow fields was introduced by the work of Hultquist [11] in 1992, whose well-known advancing-front concept has led, among other things, to the development of sophisticated methods for path surface integration in time-varying flow fields [9, 17]. The work by Krüger et al. [13] represents the most direct way of streak generation and visualization, namely by the use of large particle systems being influenced by the surrounding flow field. The absence of a triangulated mesh does however limit these point sets to discontinuous representations. Operations such as surface intersections are not possible without further effort. Funck et al. [8], Cuntz et al. [6], and Weiskopf et al. [19] introduced work on smoke surfaces and particle level set advection, focusing on the visualization of non-adaptive streak surface like structures. The first of these papers triangulates the particle system for visualization purposes and is therefore closer to our method, as far as visualization techniques are concerned.
Examples for MLS based surface approximation in the context of surface mesh reconstruction and refinement from point clouds are given by the work of Alexa et al. [1] and Mederos et al. [14].
Delaunay type mesh refinement of static point sets has been the topic of multiple papers such

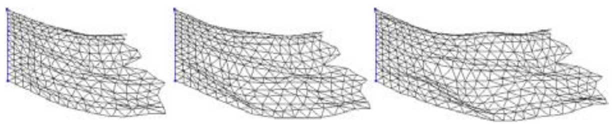


Figure 1 Triangulations of a non-adaptive streak surface in three consecutive time steps.

as the work of Chew et al. [5] and Chen et al. [4]. Both approaches use incremental mesh construction by either building a constrained Delaunay triangulation or by transforming common Delaunay algorithms to a new parametric space. Contrary to this work, the nature of our problem allows us to make use of an existing time-varying triangulation, thus eliminating the need for complete mesh reconstruction and facilitating the incorporation of multiple Delaunay triangulations into the particle insertion process, as discussed in section 5.4.

3 Streak Surface Definition

A streak surface is the locus of a set of (connected) particles that are advected by a time-dependent flow field $f : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^3$. The integral surface (1) defines a streak surface S at time t with particles emerging from points $c(s)$ at an univariate seeding curve $c : [0,1] \to \mathbb{R}^3$. Individual instances of particles are identified by their age parameter $r \in [0,t]$.

$$S(r,s,t) = c(s) + \int_{t-r}^{t} f(S(r-(t-x),s,x),x)dx\,. \tag{1}$$

In contrast to stream surfaces, streak surfaces are generally no longer tangential to the flow field and need to be updated or refined at their whole range during integration. Due to the additional parameter r, streak surfaces describe a truly three-dimensional complex in time and space, whereas stream surfaces are only two-dimensional in space. This increase of complexity prevents the use of classic approaches to adaptivity such as the concept introduced by Hultquist.
Figure 1 illustrates a simple example of (1) with a sequence of triangulations of three consecutive time steps of a streak surface. At the curve c shown in blue, 15 particles are seeded at equidistant positions. The problem exhibited by non-adaptive approaches can be identified as the far too uniform particle distribution, preventing the accurate and smooth representation of folds. One has to note, that in the discrete case a single streak surface consists of a number of consecutive static surfaces obtained in different time steps of the surface. Depending of the time resolution of the data set, coherency between consecutive surfaces might be low, producing a rough, jagged animation during rendering of the streak surface.

4 Moving Least Squares Approximation

Least Squares fitting is a common approach to data approximation, providing a method to construct functions that minimize the squared distance to a given set of data points (x_i, f_i). If a local approximation of the data is desired, the classic Least Squares scheme may be generalized to the Weighted Least Squares method by the introduction of a weighting function ω, see [16]. A polynomial function f with given degree at a point of evaluation x in a Weighted Least Squares sense is defined by (2).

$$\sum_i \omega(x,x_i)||f(x_i) - f_i||^2 \to \min\,. \tag{2}$$

The scheme obtained from moving the Weighted Least Squares over the domain of the data set to yield a continuous approximation of the field, is called *Moving Least Squares* approximation. Solving (2) for the coefficient vector a_x of a polynomial $f(x) = a_x^T \cdot b(x)$ with a given base vector $b(x)$ leads to a linear system of equations (LSE). For a linear, two-dimensional base vector $b(x) = (1 \text{ x y})^T$ it takes the following form:

$$\left(\sum_i \omega(x, x_i) \begin{pmatrix} 1 & \mathrm{x} & \mathrm{y} \\ \mathrm{x} & \mathrm{x}^2 & \mathrm{xy} \\ \mathrm{y} & \mathrm{xy} & \mathrm{y}^2 \end{pmatrix} \right) \cdot a_x = \sum_i \omega(x, x_i) \begin{pmatrix} 1 \\ \mathrm{x} \\ \mathrm{y} \end{pmatrix} \mathrm{f}_i \,. \tag{3}$$

Computational complexity of this LSE is drastically increased by degree or dimension elevation of domain or range of the data set. So does quadratic three-dimensional approximation already require solving a 10×10 system once per dimension of f_i.
Accuracy and level of detail in the reconstructions created by MLS are strongly dependent on the properties of the weighting function used. Common exponential weighting functions of the general form

$$\omega(x, x_i) = a \cdot e^{\frac{||x - x_i||^2}{r^2}} + b$$

for example, yield different results for a varying *smoothing length* r. Hereby, an increased smoothing length leads to less detailed but smoother reconstructions. The appropriate choice of r is detailed in the following sections whenever we make use of MLS.

5 Surface Generation

5.1 Algorithm Outline

Our algorithm to generate adaptive streak surfaces consists of five basic stages that are repeated for every time step of surface integration:
1. Generate new particles at the rake and insert them into the existing mesh
2. Concurrently advect all particles of the surface to their new positions
3. Restore the Delaunay property of the surface mesh by edge flipping
4. Determine curvature of the surface at every surface particle by MLS approximation
5. Adapt resolution by insertion and advection of new particles in current and previous time steps

These steps are described in detail in the following.

5.2 Particle Birth

Let $t \in [t_0, t_1]$ be the current time step. The basic representation of a streak surface usually consists of a set of particles that are advected through the flow field, being seeded at a predefined curve known as *rake*. Based on a user-defined resolution, we release a set of particles at equidistant positions along the rake. While the given resolution does directly influence the number of seed positions at the rake, the magnitude of the velocity field indirectly governs the number of particle rows that are generated and advected at the rake. If n particles with a seed distance of d are seeded at the rake, where the distance traveled in one time step at an arbitrary position on the rake is $l := ||f(.)|| \cdot \Delta t$, we release $m = \frac{l}{d}$ particles at every seeding position, leading to a total of $n \times m$ new particles.
Let T be a given triangulation in time step t of the particles released in $[t_0, t)$. We link the $n \times m$ new particles based on their neighborhood in parameter space (s, t) and connect this

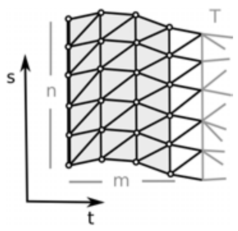


Figure 2 The triangulation of the recently seeded particles is linked to the old mesh. Note that the old mesh is still located at its position in time step $i-1$ until corresponding particles are advected.

new triangulation to the first row of particles in T, as shown in Figure 2.
Consequently we obtain a fully connected particle-based streak surface, whose triangular mesh is used for surface approximation, refinement and visualization of the surface, as explained in the following sections. As a result, particles on a streak surface do not only carry spatial information, but provide data about the (s,t) parametrization of the surface, that can be used for the generation of texture coordinates, as well as normal and neighborhood information. It is important to point out, that the basic definition, computation and visualization of a streak surface does not require the availability of connectivity information between particles. As described in the next sections, the triangulation created during particle birth is an auxiliary construct to reduce the computational effort for surface approximation and particle density calculations. While it only represents a linear approximation of the true streak surface, it can additionally be used for basic surface visualization.

5.3 Particle Advection

Particle data of time step $t-1$ has to be propagated to the new time step t and particles need to be advected to their respective new positions in t according to the appropriate velocity values obtained from vector field evaluation. To compensate possible artifacts in poorly time resolved data sets or data that has a high curvature in time, we take special care during vector field approximation. While adaptive Runge-Kutta approximation schemes can determine the step size or order of integration k used during particle advection, the necessary interpolation between adjacent time steps that needs to be performed, if $k>1$ often introduces artifacts if the degree of interpolation is too low, as in (4). Higher order integration such as cubic Hermite interpolation, taking into account $f(p,t-2)$, $f(p,t-1)$, $f(p,t)$, and $f(p,t+1)$ for particle advection from $t-1$ to t double the number of required field evaluations but tend to yield more accurate surfaces.

$$f(p,t-1+\frac{j}{k}) = \left(1-\frac{j}{k}\right) f(p,t-1) + \frac{j}{k} f(p,t) \tag{4}$$

where $f(p,t-1+\frac{j}{k})$ is the velocity of a particle with position p and integration order k during the j-th step of advection from $t-1$ to t. This scheme subdivides the time interval $[t-1,t]$ into k time intervals with linearly interpolated velocity fields. To reach a sufficient accuracy, we determine k for every particle individually by comparison of angular deviation between velocity vectors resulting from consecutive vector field evaluations. This particle advection scheme reduces path deviations in data sets with large time steps and improves visual coherence. A mapping of the order of integration k onto a streak surface is shown in Figure 3. As individual particles from surfaces of consecutive time steps are matched in our

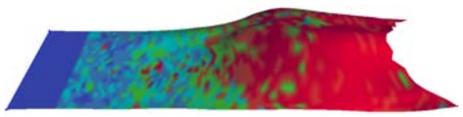


Figure 3 A simple streak surface is color-mapped with the integration order $k \in [1, 16]$. Order of integration is mapped to the hue spectrum from 0° to 240°, with red being the maximum. As can be seen, even adjacent particles might need a highly different number of vector field evaluations.

data structure, storing not only the final position of an advected particle, but reusing the k intermediate ones obtained from field integration during streak surface visualization facilitates the rendering of smooth surface animations even in data sets with low time resolution. In these computations, the magnitude of the smoothing length r of the weighting function used in MLS during vector field evaluation is inversely related to the point density of the data set. This point density is usually either an output value of the CFD simulation itself, or has to be determined on the fly by k-nearest neighbor computations or similar methods. We use MLS for vector field approximation because of its independence of a computational grid and because it is used as interpolant by the simulation that generated our test data sets.
To speed up the advection process, particle locality is used for efficient data set caching and parallel particle advection. We impose a rectangular grid on the surface clustering particles into independent sets and delegate the according computations from (4) to different CPU cores.
If a particle of the surface leaves the boundaries of the data set during advection, the particle itself and adjacent mesh elements are deleted, efficiently trimming the streak surface.

5.4 Delaunay Meshing

As mentioned before, the increased dimensionality of streak surfaces has a direct influence on the adaptivity methods that can be used. Hultquist's approach of inserting stream lines at a single curve-like consistent front cannot be generalized to the insertion of streak lines at a one-dimensional front of the streak surface. The two-dimensional character of the generalized front definition for streak lines requires the insertion of refinement structures on arbitrary positions of the surface. This fact virtually rules out the exclusive refinement along time- and streak lines of the surface, as most interior refinement points would miss these grid lines. Thus, we use a Delaunay type progressive mesh to define criteria for particle insertion.
Various work on the construction of Delaunay type surfaces from point clouds has been published over the years. For example, Gopi et al. [10] use a projective local Delaunay mesh for surface reconstruction. The underlying particle concept of streak surfaces suggests the use of point cloud reconstruction methods to obtain a surface mesh. However, the evolving property of the streak-surface mesh facilitates topologically correct (re-) meshing in a certain time step based on connectivity information given by prior time steps as well as triangle, edge, and node matching over multiple time steps.
The mesh structure of the previous time step generally loses its Delaunay properties as particles are advected, if corners of adjacent triangles describe different paths through the flow field, as illustrated in Figure 4. Since we want to estimate particle distributions using circumcircle properties for mesh refinement in a later step of the algorithm, it is important to have a well conditioned triangulation avoiding skinny triangles. Therefore, we choose to impose Delaunay's mesh properties on the triangulation of our surface, as the minimal circumradius property is a direct indicator of particle density. The availability of a mesh on the current particle set greatly simplifies construction of a curved Delaunay mesh, since

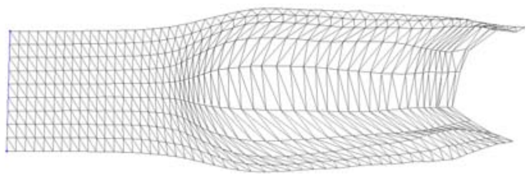


Figure 4 A uniform grid of a non-adaptive streak surface gets deformed, producing ill-conditioned triangles of bad aspect ratios.

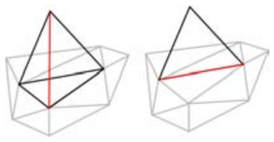


Figure 5 Edges of a tetrahedral structure on the curved surface mesh cannot be flipped without changing the underlying topology and creating holes.

this fact makes it possible to use local edge flipping instead of global mesh construction algorithms such as line-sweep. Moreover, reusing the previous particle connectivity allows matching of non-flipped edges as well as triangles of different time steps.

An edge of a mesh is *flippable*, if it is shared by two triangles and its flipped counterpart is not already part of the mesh , see also [7]. Let $e = (b, c)$ be a flippable edge shared by the two triangles $\Delta_1 = (a, b, c)$ and $\Delta_2 = (d, c, b)$. We flip e, if the sum of the angles α at a and β at d exceeds π, resulting in $\Delta'_1 = (a, d, b)$ and $\Delta'_2 = (c, d, a)$, satisfying local Delaunay properties. This flipping procedure, having been shown to converge for curved surfaces by Dyer et al. [7], is repeated until the mesh contains no more flippable edges. If non-flippable edges of tetrahedral structures of the curved triangle mesh remain after all flippable edges have been swapped, see Figure 5, we collapse the corresponding tetrahedra by removing the particle at its tip. Such tetrahedra generally represent noise in the form that they indicate a particle that evades the path of the streak surface and has earlier been inserted at a wrong position. After this step, our surface mesh is usually Delaunay conform, with exceptions to rare special cases of non-flippable edges. We avoid insertion of additional particles on the surface to obtain a mesh that fully satisfies Delaunay properties as proposed in [7] to reduce the resulting computational overhead that is needed to advect the new particles through the vector field. In these special cases we allow our mesh to be locally non Delaunay, as the according triangles are commonly not badly shaped due to the fact of mesh deformation of large parts of the surface in every step of integration.

5.5 Curvature Approximation

We determine local geometric complexity of the streak surface by measuring its local curvature. Curvature at a point p of a bivariate surface S is described by the maximal and minimal curvature of the curves on S that result from intersecting S with planes through p containing the surface-normal vector at p. These curvature values are called *principal curvatures* and will be used in the following as an indicator of whether to refine the mesh of the surface.
For adaptive particle insertion we need to know both a parametric form of the surface, as well as its local curvature at every particle of the surface. Instead of handling these tasks by two different approximation techniques, we use one weighted Least Squares approximation to both obtain a valid surface representation as well as to calculate the according local

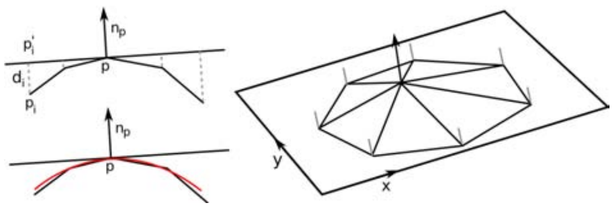


Figure 6 Two-dimensional illustration of a MLS curve approximation, MLS curve is shown in red (left) and three-dimensional analogon with a particle neighborhood-level of one (right).

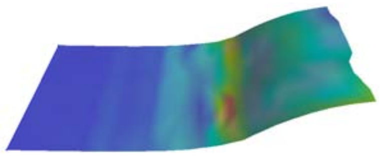


Figure 7 Maximum absolute curvature as hue color-map on a streak surface. Red color shows regions of high curvature, blue indicates planar sections.

curvature. We compute local surface approximation at a particle p based on particle offsets from a local tangential plane with vertex normal n_p as shown in Figure 6, resulting in a new set of projected data points $(p_i' = (x_i, y_i), d_i)$ that are approximated by a scalar-valued bivariate quadratic MLS. If the surface is to be approximated at a particle p, we assemble surrounding particles p_i by a neighborhood search along the edges of the mesh up to a neighborhood distance of two, as the bivariate quadratic LSE that has to be solved for a MLS approximation requires the data of at least six non-collinear points. The smoothing length r of the weighting function used during MLS approximation is chosen in a way, such that $w(p, p_j') = \epsilon$ with p_j' being the neighboring particle that is farthest away from p. This MLS surface representation defines every point on a surface by its orthogonal distance to the tangential plane.
The eigenvalues of the Hessian of this polynomial approximation represent the principal curvatures c_1 and c_2. The Hessian of a bivariate scalar function takes the form of a 2×2 matrix, which can easily be calculated using a parametric form of the approximating MLS polynomial [12]. We use the *maximum absolute curvature* $c = \max(||c_1||, ||c_2||)$ as a measure of local feature size, which has proven to lead to good results in other applications, see [2]. Figure 7 shows a simple streak surface colored according to maximum absolute curvature. While Moving Least Squares for surface approximation is computationally more expensive than to simply use the piecewise linear representation of the surface described by the mesh itself, it both reduces errors in the particle insertion step, as explained in the next section and gives a more accurate notion of the surface curvature, by filtering small noise due to its approximating behavior.

5.6 Particle Insertion

The general notion of adaptivity is to sample a surface according to its geometric complexity, meaning that regions of high curvature need more samples to be represented accurately than regions that are almost planar. In the context of parametric or implicit surface modeling, curvature-dependent particle density control is often handled by minimization of an energy function [15]. In our case the availability of a correct coarse mesh as well as the absence of a gradient-definition requires different adaptivity measures.

The crucial step of streak surface adaptivity is the correct insertion of new particles at positions throughout the surface. The most accurate way of adaptively inserting particles is the insertion of a particle in the appropriate first time-step t_0, whenever an ill-conditioned particle-density is detected at an arbitrary later time step t_1. This method does however require the computationally inefficient advection of all newly inserted particles from t_0 til t_1. We therefore prefer to insert new particles directly into the evaluated time step or few of its predecessors.

Given the computed curvature at a particle p, see section 5.5, we are able to decide whether to insert new particles in the immediate neighborhood of p. Figure 4 demonstrates, why refinement purely along time- and path lines is not desired in the context of streak surfaces, since right angles between such lines are not maintained over time and such a skewed global coordinate frame is not suitable for balanced control of particle densities. We therefore use the local feature size in form of the curvature at a position p to directly describe the desired maximal allowed distance between two neighboring particles:

$$r_c = \frac{b}{\max(||c_1||, ||c_2||)}$$

where b controls the impact of curvature on the degree of mesh refinement and is commonly chosen be a value between zero and one. Circumradii of all triangles adjacent to p are compared to r_c, if a circumradius exceeds this threshold, and is larger than a pre-defined minimum triangle size, a new particle is inserted on the according triangle. We commonly limit the minimum triangle size to a size a few magnitudes smaller than the data set resolution to avoid numerical instabilities and oversampling. As our underlying mesh is mostly Delaunay conform and therefore satisfies the smallest circumcircle property, the circumradius of triangles is a valid measurement of particle density at a specific region.

Particle insertion itself poses the question of how to find the optimal position of insertion. On planar Delaunay meshes, one would choose the circumcenter of a triangle as location and handle point insertion in a simple way: Remove all triangles, whose current circumcircle includes the newly placed point and connect vertices of the resulting polygon with the new point. The resulting mesh again satisfies the Delaunay property. However, on curved surfaces this method yields several problems. Location of the circumcenter of obtuse triangles is not trivial, as the circumcenter is located outside of the triangle, thus requiring a search on neighboring triangles and even performing intersection operations, if the surface bends. Moreover, the point-in-circumcircle property does not work as expected from the planar case for both the common projected-circumcircle and the circumsphere definition on curved surfaces, if the point is not inserted on the piecewise linear representation of the surface. These considerations motivate our approach of particle insertion.

Once a triangle with a too large circumradius is detected, the according new particle is either inserted at the centroid of this triangle or on one of its edges. More precisely, we subdivide the longest edge of the triangle if the circumcenter is located outside of the triangle or the triangle itself is a boundary element of the surface. The problem caused by an invalid insertion of particles at the centroid of a boundary triangle is depicted in Figure 8, where the quality of the aspect-ratio of the boundary triangle is degraded significantly. These two types of particle insertion lead to well conditioned triangles with smaller circumcircle size as soon as the Delaunay property of the mesh is restored in the mesh at the next time step.

This insertion scheme yields adaptive streak surfaces but tends to introduce errors in the particle insertion positions that lead to further surface deviations during advection, even if newly inserted particles are offset according to the local MLS approximation of the surface as computed in the previous step of our algorithm. To avoid these artifacts, we

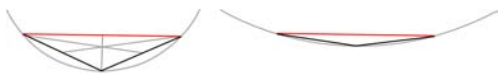
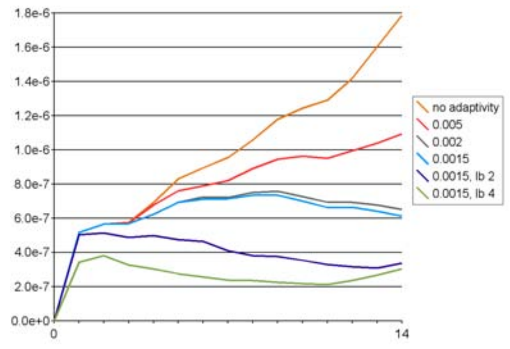


Figure 8 Circumradius of the triangle adjacent to the boundary edge (red) increases after particle insertion, as the boundary edge cannot be flipped to produce better conditioned triangles. Such cases are avoided by splitting of boundary edges.

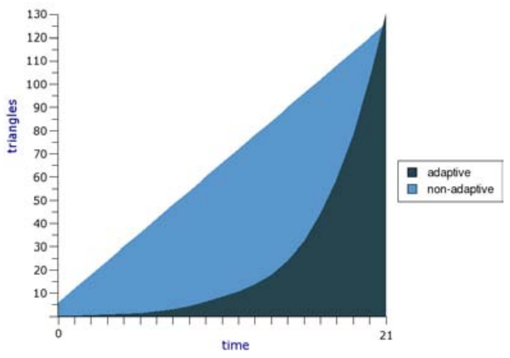
introduce the concept of looking back at a history of $lb > 0$ previous instances of concerned triangles or edges. If a triangle (edge) tri_1 is bound to be refined in time step t_1, we find matching triangles (edges) in time steps $t_i \in [t_1 - lb + 1, t_1 - 1]$ and insert particles into according triangles in t_i, until either step $t_1 - lb + 1$ is reached, the curvature of the surface at the corresponding position in t_i falls below a given minimum, or no matched preceding triangle exists and cannot be created by edge flips. These m newly inserted particles describe an ordered sequential set of points $[p_0, p_{m-1}]$ located on the linear representation of the individual surfaces $[S_{t_1-m}, S_{t_1}]$. As these particles need to lie on the same path line, positions of particles p_i, with $0 < i <= m - 1$ are offset according to the position of p_{i-1} after one full time step of particle advection.
Our tests have shown, that this leads to a great reduction of noise artifacts in later time steps, that are caused by deviating paths of particles that were inserted at poorly approximated positions.
Curvature as well as circumradius properties can be used for the removal of particles as well. In the case, that a particle p has a sufficiently small curvature, i.e. the surface is almost planar, and all adjacent triangles have small circumradii, it is valid to remove the concerned particle without losing any details in the surface representation. In realistic applications such as the test data sets shown in section 6 such situations do hardly occur.

6 Results

In the following we present numerical examples of adaptive streak surface integration in different data sets generated by a point based, grid-less CFD simulation [18]. The first data set consists of about 25.000 particles simulating flow around a cylindrical obstacle with ellipsoidal profile. Dimensions of the obstacle as well as velocity of the fluid were chosen specifically to yield a high Reynolds number, leading to a three-dimensional Von Kármán vortex street and a low time resolution. The resulting disturbed flow structures have optimal properties to observe the quality of our adaptive streak surface integration approach. Figure 10 depicts six non-consecutive time steps of adaptive streak surface integration in this first data set. Near the vortex structures, the surface mesh has a high curvature and needs to be sampled accordingly, as described in sections 5.5 and 5.6. Highly twisted and folded surface structures in the Von Kármán vortex street demonstrate the robust refinement of our approach. The surface in the last time step shown consists of around 110.000 particles. The second data set contains a spherical obstacle, approx. 26.000 flow particles and fluid flow with a Reynolds number close to the one from data set one. Both data sets have a low resolution in the time dimension to demonstrate the robustness of artifact avoidance by utilization of our look back strategy. In contrast to the first data set, absence of a distinguished direction on the obstacle leads to aperiodic, intensively folded flow structures, see Figure 11. To give an impression of how accurately surfaces generated by the adaptive integration scheme and the Delaunay mesh refinement method introduced in this work represent the "real" surface, we show comparisons between high-resolution non-adaptive meshes and adaptive surfaces in the two Figures 14 and 15. The former illustrates how the



(a)



(b)

Figure 9 (a) Error statistics for different levels of adaptivity and look back. **(b)** Chart of the number of triangles (in multiples of 1.000) in the first 22 time steps of the streak surface shown in Figure 10. Dark blue refers to the adaptive streak surface, light blue to the non-adaptive streak surface, that has an approximately identical number of triangles at step 22.

Table 1 Exemplary measurements of total streak surface generation times in milliseconds for 25 time steps of data set one with 20× parallel particle advection. Final particle number is approx 40.000 for both approaches.

Method	Advection	Delaunay	Adaptivity
non-adaptive	1.768.470	2.865	0
adaptive, 0.002, 2 lb	790.890	1.155	9.081

folds of a high-resolution mesh are correctly refined and represented by an adaptive surface with only half as many triangles as an equivalent non-adaptive surface. Efficient distribution of particle densities can be observed in Figure 15, where the adaptive version of a streak surface clearly has a particle distribution of better quality than a non-adaptive surface with an equal number of points.
Statistical error measurements are shown in Figure 9a. The measured two-sided surface mesh error is computed with respect to (5).

$$E(S_1, S_2) = \frac{1}{|P_1| + |P_2|} \left(\sum_{p \in P_1} d(p, S_2) + \sum_{q \in P_2} d(q, S_1) \right) \tag{5}$$

where S_1 and S_2 are the meshes of two streak surfaces at the same time step with according point sets P_1 and P_2. It is important to note that, as d computes the minimal distance of a point to a surface mesh, error values for the same point sets usually differ if the mesh is changed. While small features in the error curves of Figure 9a may therefore be caused by flipped edges, the overall benefit gained from reducing the minimal surface resolution and increasing the number of look back steps is clearly visible. The plotted exemplary measurements were taken at different time steps of a data set that was scaled to fit into a unit cube and represent absolute per particle error when comparing the according surface to a high resolution ground truth streak surface. The shown measurements represent 15 consecutive time-steps of a surface evolving under extreme stretching and turbulence conditions - a representative scenario that requires reliable surface reconstruction techniques to produce accurate streak surfaces.

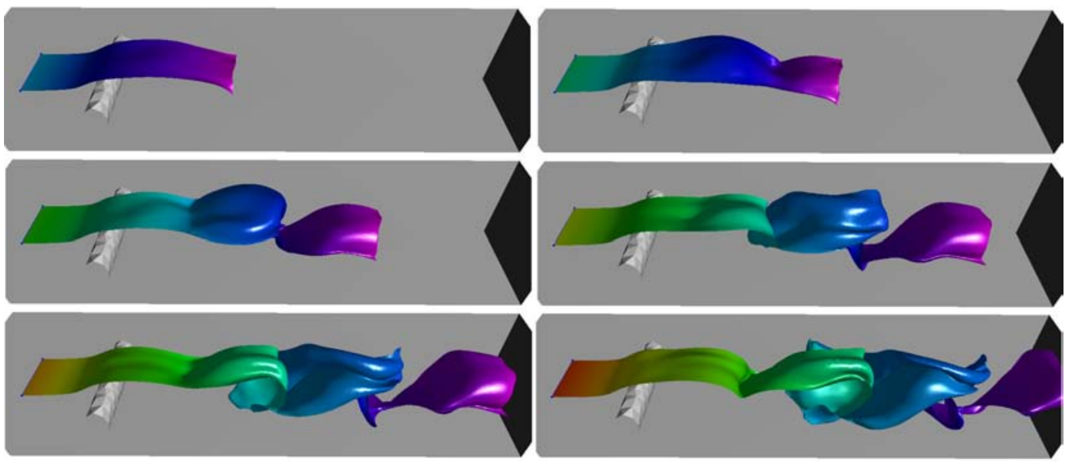


Figure 10 A sequence of six time steps in a three-dimensional Von Kármán vortex street produced by test data set number one. The extracted surface is colored with respect to time, with purple showing the particles generated at time t_0 and red being used for new particles. Details of twisting folding, and stretching are visible.

In addition to the direct impact of our adaptivity approach on particle counts and distributions, the reduced number of particles does influence the computation times of surfaces as well. We show a representative graph of particle number development in a surface generated in the ellipsoidal data set in Figure 9b. The triangle count for the adaptive surface shows exponential growth as long as the surface is completely inside the domain of the data set and is not trimmed. Exponential growth is stopped as soon as parts of the surface leave the data set. Integrals of the depicted curves are directly proportional to the time spent for total particle advection. From the given graph, it is clearly visible that adaptive streak surfaces require the advection of a larger number of particles to reach the same final number. Due to better distribution of particle densities, an adaptive surface with the same number of particles generally represents a better approximation of the real surface. The previously mentioned assumption about particle advection time is verified by measurements during surface integration. The total time spent for adaptivity measures is a combination of computation times for surface and curvature approximation as well as particle insertion. In our tests only a percentage of less than 2% of the total surface generation time was consumed by the adaptivity method, even if particle advection was performed 20 times in parallel. As the adaptive integration scheme discussed in section 5.3 commonly requires multiple evaluations of the vector field for every particle, the amount of time dedicated to particle advection is much higher than the one used up by mesh refinement, thus noticeably speeding up the integration process. Further improvements on this performance can be made by parallelization of the mesh refinement algorithm. While in our tests generation of adaptive surfaces was in average 4 times faster than creation of equivalent higher resolved surfaces, when using 10× parallel particle advection, it is difficult to obtain meaningful absolute time comparisons (cf. Table 1), as speed-up is highly dependent on the method of vector field approximation and flow complexity. In general, the relative speed-up gained by adaptive surface generation is proportional to the complexity of flow and field evaluation methods.

For insightful visualization of generated surfaces, we utilize several known visualization techniques. Four of these techniques are displayed in Figure 12. While the first picture showing a standard particle based visualization of the surface is not capable of conveying the

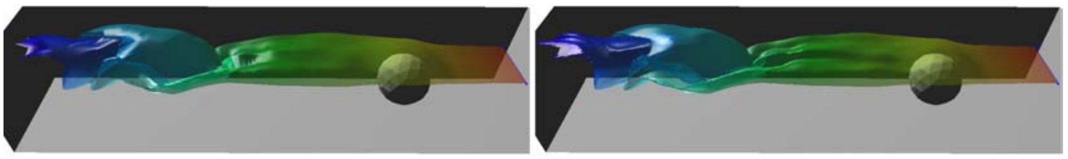


Figure 11 Transparent rendering of a non-adaptive and an adaptive streak surface with the same seeding resolution extracted from test data set number two. The spherical obstacle produces aperiodic disturbances.

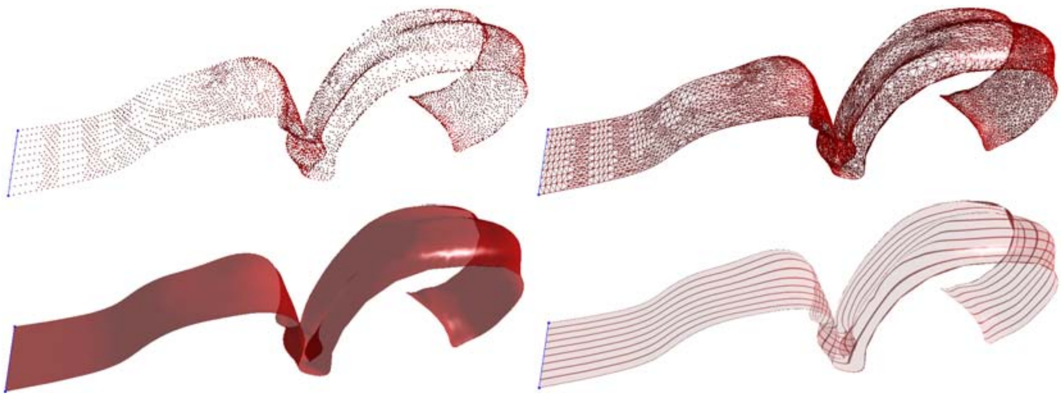


Figure 12 One time step of streak surface integration shown in four different visualization techniques for surface rendering. Techniques are: Surface particles rendered as shaded points, wire-frame triangulation, solid transparent triangulation, and texture mapping showing streak line structures.

impression of a homogeneous surface, it gives an insight into adaptive particle distributions. Particle density and connectivity information are shown by a direct wire-frame representation of the Delaunay construct as used in the second frame. Shading and texturing techniques are applied to the solid Delaunay triangulation based surface visualizations. We use a simple axis-based triangle pre-sorting approach for transparent surface rendering as shown in the last two frames. Texturing allows the rendering of streak- or time-line like parts of the surface, former is shown in frame four.

These renderings of streak surfaces have in common, that the Delaunay construct used for density estimation is used directly for surface visualization, leading to a simple and fast visualization allowing the analysis of adaptivity properties. For high quality surface rendering, one would typically use a smooth interpolation of the surface point set and a much finer resolved triangle grid to get rid of fine discontinuities of the simple linear representation.

To illustrate the use of the methods introduced in this paper in a real application, we present an example of an adaptive evolving mesh being equivalent to a streak surface without a seeding rake in Figure 13. The application is concerned with stirring of a fluid at high temperature and consists of a cylindrical barrel and two rotating wheels with four attached mixing poles. The extracted mesh segments the fluid in a stirring simulation into two separate volumes, giving an impression of how our approach to adaptive streak surfaces can be used to separate regions of different flow as done in vector field topology.

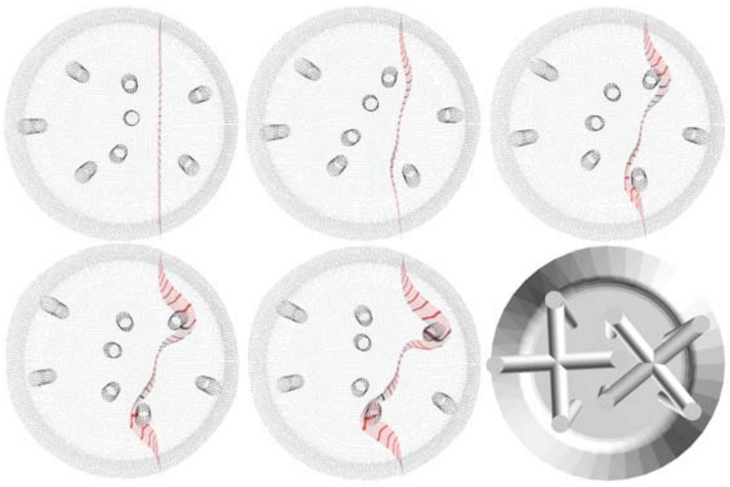


Figure 13 Refinement mesh in a stirring simulation.

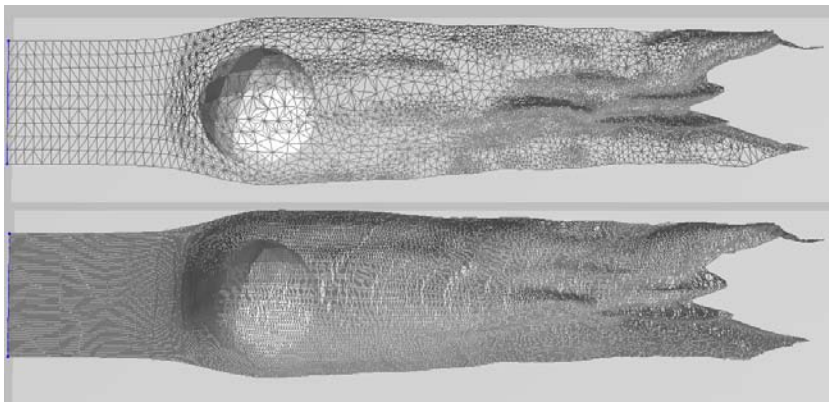


Figure 14 Triangulations of an adaptive (top) and a high-resolution non-adaptive streak surface passing a spherical obstacle. Triangle counts in the time step shown evaluate to approximately 20.000 and 45.000. The adaptive surface yields a highly diverse particle density, while accurately representing fold-like features.

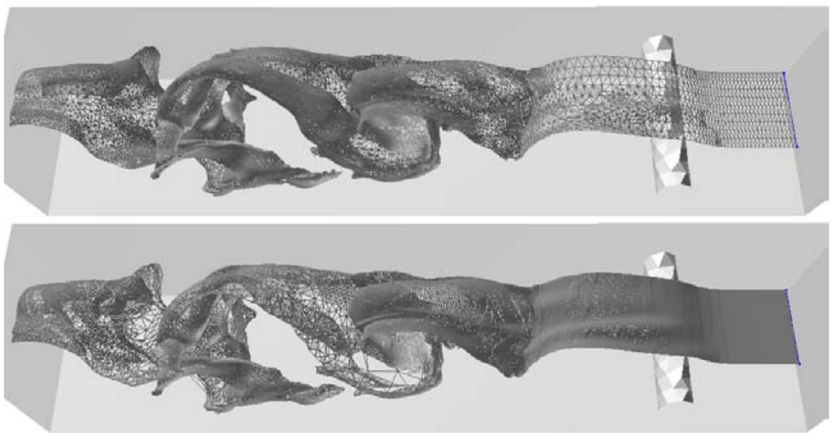


Figure 15 Triangulations of an adaptive (top) and a non-adaptive streak surface with approximately 110.000 triangles each. The non-adaptive version shows less optimal particle distributions and regions with low mesh resolution.

7 Conclusion

In this paper we have introduced an approach to fully adaptive streak surface integration. Our method is based on MLS approximation for the evaluation of the vector field and surface as well as curvature approximation. For particle insertion we utilized the circumradius properties of a Delaunay type evolving mesh, which is restructured after every time step to restore Delaunay properties. The distinction between two types of particle location during insertion avoid the generation of triangles with bad aspect ratios. A look back strategy during particle insertion further reduces approximation errors.
The results shown in section 6 demonstrate the accurate, robust, and fast integration of adaptive streak surfaces generated by our method. Moreover we have presented results in a real world application of a mixing process, illustrating the suitability for practical flow analysis and topology-based visualization.
The methods introduced in this work are portable to other fields of visualization where moving or evolving meshes are concerned, such as the concept of unsteady flow volumes presented by Becker et al. [3]. Common approaches to the extraction of vector field topology of fluid flows [20] based on the use of stream surfaces as three-dimensional separatrices in the stationary case can be generalized to vector field topology methods in non-stationary fields by the application of streak surfaces for segmentation of time-varying volumes.
Future work on the topic of adaptive streak surface construction may include the utilization of improved spatial clustering for further parallelization of the surface generation process as well as the integration of vector field singularities into the streak surface generation process.

Acknowledgments

This work was supported by the German Research Foundation (IRTG 1131) and by the Center for Mathematical and Computational Modeling (CM)^2.

References

1 Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishmann, David Levin, Claudio T. Silva: Point Set Surfaces *Visualization '01, pp. 21–28 (2001)*

2 Bruno Rodrigues de Araujo, Joaquim Armando Pires Jorge: Curvature Dependent Polygonization of Implicit Surfaces *Proceedings of the Computer Graphics and Image Processing 2004, pp. 266–273 (2004)*

3 Barry G. Becker, Nelson L. Max, David A. Lane: Unsteady Flow Volumes *Proceedings of the 6th conference on Visualization '95, p. 329 (1995)*

4 Hao Chen, Jonathan Bishop: Delaunay Triangulation for Curved Surfaces *6th International Meshing Roundtable Proceedings, pp. 115–127 (1997)*

5 L. Paul Chew: *Guaranteed-Quality Mesh Generation for Curved Surfaces 9th Symposium on Computational Geometry, pp. 274–280 (1993)*

6 Nicolas Cuntz, Andreas Kolb, Robert Strzdka, Daniel Weiskopf: Particle Level Set Advection for the Interactive Visualization of Unsteady 3D Flow *Eurographics/IEEE VGTC Symposium on Visualization 2008*

7 Ramsay Dyer, Hao Zhang, Torsten Möller: Delaunay Mesh Construction. *Eurographics Symposium on Geometry Processing, pp. 273–282 (2007)*

8 Wolfram von Funck, Tino Weinkauf, Holger Theisel, Hans-Peter Seidel: Smoke Surfaces: An Interactive Flow Visualization Technique Inspired by Real-World Flow Experiments *IEEE TVCG 2008, pp. 1396–1403 (2008)*

9 Christoph Garth, Hari Krishnan, Xavier Tricoche, Tom Bobach, Kenneth I. Joy: Generation of Accurate Integral Surfaces in Time-Dependent Vector Fields *IEEE TVCG 2008, pp. 1404–1411 (2008)*

10 M. Gopi, S. Krishnan, C. T. Silva: Surface Reconstruction Based on Lower Dimensional Localized Delaunay Triangulation *Computer Graphics Forum 19, (2000)*

11 J. P. M. Hultquist: Constructing stream surfaces in steady 3D vector fields *Visualization '92, pp. 173–175 (1992)*

12 Soo-Kyun Kim, Chang-Hun Kim: Finding ridges and valleys in a discrete surface using a modified MLS approximation *Computer-Aided Design, Vol. 38, No. 2, pp. 173–180 (2006)*

13 Jens Krüger, Peter Kipfer, Polina Kondratieva, Rüdiger Westermann: A Particle System for Interactive Visualization of 3D Flows *IEEE TVCG 2005*

14 Boris Mederos, Luiz Velho, Luiz Henrique de Figueiredo: Moving Least Squares Multiresolution Surface Approximation *Proceedings of the Computer Graphics and Image Processing 2003, pp. 19–26 (2003)*

15 Miriah D. Meyer, Pierre Georgel, Ross T. Whitaker: Robust Particle Systems for Curvature Dependent Sampling of Implicit Surfaces *Proceedings of the International Conference on Shape Modeling and Applications 2005, pp. 124–133 (2005)*

16 A. Nealen: An As-Short-As-Possible Introduction to the Least Squares, Weighted Least Squares and Moving Least Squares Methods for Scattered Data Approximation and Interpolation, *Discrete Geometric Modeling Group, TU Darmstadt (2004)*

17 Tobias Schafhitzel, Eduardo Tejada, Daniel Weiskopf, Thomas Ertl: Point-Based Stream Surfaces and Path Surfaces *Graphics Interface 2007, pp. 289–296 (2007)*

18 S. Tiwari, J. Kuhnert: Finite Pointset method based on the projection method for simulations of the incompressible Navier-Stokes equations. *Springer Lecture Notes in Computational Science and Engineering: Meshfree Methods for Partial Differential Equations I (2001)*

19 Daniel Weiskopf: Dye advection without the blur: A level-set approach for texture-based visualization of unsteady flow *Proceedings of Eurographics 2004, pp. 479–488 (2004)*

20 Alexander Wiebel, Xavier Tricoche, Gerik Scheuermann: Extraction of Separation Manifolds using Topological Structures in Flow Cross Sections. *Topology-Based Methods in Visualization II, pp. 31–44 (2009)*