Eurographics Conference on Visualization (EuroVis) 2012
S. Bruckner, S. Miksch, and H. Pfister
(Guest Editors)

Volume 31 (2012), Number 3

Automatic Stream Surface Seeding:
A Feature Centered Approach

M.Edmunds[1], R.S.Laramee[1], R.Malki[1], I.Masters[1], T.N.Croft[1], G.Chen[2], E.Zhang[3]

[1]Swansea University, Wales, UK. [2]University of Utah, United States. [3]Oregon State University, United States.

Abstract

The ability to capture and visualize information within the flow poses challenges for visualizing 3D flow fields. Stream surfaces are one of many useful integration based techniques for visualizing 3D flow. However seeding integral surfaces can be challenging. Previous research generally focuses on manual placement of stream surfaces. Little attention has been given to the problem of automatic stream surface seeding. This paper introduces a novel automatic stream surface seeding strategy based on vector field clustering. It is important that the user can define and target particular characteristics of the flow. Our framework provides this ability. The user is able to specify different vector clustering parameters enabling a range of abstraction for the density and placement of seeding curves and their associated stream surfaces. We demonstrate the effectiveness of this automatic stream surface approach on a range of flow simulations and incorporate illustrative visualization techniques. Domain expert evaluation of the results provides valuable insight into the users requirements and effectiveness of our approach.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Algorithm/Technique— Automatic Stream Surface Seeding Data Clustering Hierarchy Data Vector Field Data Geometry-based Techniques feature-centered + overview Techniques Illustrative Visualization Flow Visualization

1. Introduction

Flow visualization is an important and powerful means for analyzing, exploring and communicating simulation or experimental results. Flow visualization results can differ in their complexity, quality and style [PVH*03] [LHZP07] [PL09]. Stream surfaces have become increasingly popular in recent years, and have important inherent characteristics that can enhance the visual perception of complex flow structures [MLP*10]. Lighting and shading reinforce the perception of shape and depth, images or textures can be mapped to the surface primitives providing additional visual information, color and transparency can be used to convey additional data attributes.

Surfaces in general are able to not only capture the features within the flow, but also have the inherent ability to convey further information about the local attributes of the flow [LGD*05]. This combined with the reduction in visual clutter when compared to using glyphs or streamlines, significantly enhances the utility of surfaces for practitioners.

A stream surface is the integration of a one dimensional curve through 3D steady flow. The resulting surface is everywhere tangent to the local flow. Since there is no normal component of the velocity along stream surfaces, thus they are useful for separating distinct regions of similar flow behavior. In practical applications [Hul92] [GTS*04] a discretized approximation of the stream surface is constructed by integrating discretized seeding curves through the vector field.

Stream surfaces for visualization face many challenges. These surfaces must represent an accurate approximation of the underlying simulation. Adequate sampling must be maintained while reducing the unnecessary computational overhead associated with over-sampling [GKT*08]. When using surfaces the problem of occlusion arises. This may stem from multiple surfaces that occlude one another, a large surface that results in self occlusion, or a combination of both. A general solution to this problem is to use transparency. With stream surfaces we have additional options. Illustrative techniques [BWF*10] [HGH*10] can be used to improve perception.

Much work has been conducted focusing on construction and illustration techniques for stream surfaces, however, not all stream surfaces are of equal importance to domain experts. Exploration of the space of all possible stream surfaces manually, as done in most existing frameworks, is often impractical for the user [LWSH04] [LGD*05]. This interaction is based on trial and error, as the user must manually refine the initial seeding location, orientation, and size iteratively. This can be error prone and time consuming. In this paper, we make one of the first attempts to provide an automatic stream surface seeding algorithm, which we hope will help relieve the amount of work by the domain experts and therefore make stream surfaces a more popular visualization technique in practice.

Stream surfaces must be seeded such that they capture important characteristics of the flow. Seeding curves must be positioned in the neighborhood of structures best representing the flow field. Stream surface orientation is an important consideration in light of these requirements. Of the possible visualization techniques that can be used to simplify the flow field and present potential seeding locations, vector field clustering algorithms and feature extraction techniques are considered.

Explicit Feature Extraction vs. Vector Field Clustering. Explicit feature extraction techniques perform a search of the flow field in order to locate and visualize specific sub-sets of the flow [PVH*03]. However, features are not always well defined. For example, there is no universal definition of a vortex. Features are often user-dependent and thus they may not always be predicted a priori [LGD*05]. Each type of feature requires a special algorithm to extract it. Implementation of an algorithm for each type of feature may not be practical. Many explicit feature extraction algorithms rely on a threshold value(s) which ultimately determines the presence (or absence) of the feature of interest. This can result in false positives or missing sub-sets of interest (this is especially true for 3D CFD data exploration). In contrast, vector field clustering does not rely on an algorithm threshold value and can highlight weak or

partial features in the flow. Vector field clustering has been used to show interesting flow features for real world data sets [PGL*12].

Vector field clustering algorithms provide a general approach to generating a simplified, feature-based representation of vector fields. We employ this approach in our framework of automatic placement of the seeding curves. Clustering distance measures can be tuned to yield a range of results. They offer the advantage of presenting a detailed representation in important areas of the flow field, and offer flexibility required by the user. General clustering algorithms are based on agglomerative hierarchical grouping techniques which are widely used in the information visualization domain [XI05].

In this paper, we present a novel adaptation of a vector field clustering algorithm that can be guided by the user in order to automatically seed stream surfaces. The clustering technique, based on an error driven distance measure, is used to locate potential stream surface seeding positions which are used to generate insightful feature based representations of the flow field.
The main benefits and contributions of this paper are:

- An adaption of vector field clustering to guide stream surface seeding.
- An approach to automatically locating seeding positions associated with important structures within the 3D flow field.
- A technique for generating seeding curves automatically which reflect important characteristics of the flow field.
- A technique combining flow curvature with illustrative techniques to provide enhancements to the perception of the visualization.

We illustrate how to capture the characteristic structures within the flow field such as rotational flow. The key to capturing the required properties of the flow is in defining the appropriate clustering metric and providing the user with the flexibility to guide the seeding within the domain. Our focus pays particular attention to the flexibility of the clustering technique combined with the seeding curve generation strategies.

The rest of this paper is divided into the following sections: A review of related literature is conducted in section 2. A detailed presentation of the algorithm is given in section 3. The results are reviewed in section 4. A domain expert evaluation is provided in section 5. Conclusions and future work are discussed in section 6.

2. Related Work

The related work falls into the following subsections: streamline seeding and placement strategies, surface placement strategies, rendering and illustration techniques. We refer the readers to McLoughlin et. al. [MLP*10] for a complete overview of flow visualization literature.

Streamline Seeding Strategies Turk and Banks [TB96] present a method for the image guided placement of streamlines. Zöckler et al. introduce a method of illuminating streamlines [ZSH96]. For the placement of streamlines a stochastic seeding algorithm is applied. See Weinkauf et al. [WHN*03] [WT02] for applications of this seeding strategy. Mattausch et al. [MT*03] combine the illuminated streamlines technique of [ZSH96] with an extension of the evenly-spaced streamlines seeding strategy of Jobard and Lefer [JL97] to 3D.

Ye et al. use templates at critical points to effectively position streamlines in the flow field [YKP05]. This is followed by Poisson seeding in empty regions, and then filtering using geometric and spacial attributes. Chen et al. [CCK07] present a novel method for the placement of streamlines based on a similarity method which compares candidate streamlines incorporating their shape and direction as well as their Euclidean distance from one another. Li et al. [LS07] present a streamline placement strategy for 3D vector fields. This is the only approach of its kind where an image-based seeding strategy is used for 3D flow visualization. Interactive seeding strategies have been used in various, real-world applications including the investigation and visualization of engine simulation data [Lar02] [LWSH04] [LGD*05].

An image-space-based method for placement of evenly-spaced streamlines on boundary surfaces is presented by Spencer et al. [SLCZ09]. The complexity of tracing in the large unstructured grids that typically result from CFD simulations is avoided. Streamline density is controlled by an adaption of the method of [JL97]. Delmarcelle and Hesselink [DH93] visualize tensor fields using hyperstreamlines. The approach by Vilanova et al. [VBvP04] concentrates on seeding hyper-streamlines and the use of surfaces to visualize Diffusion Tensor Imaging data. A user study of 3D vector field visualization methods is conducted by Forsberg et al. [FCL09] More recently Marchesin et al. [MCHM10] present a view-dependent strategy for seeding streamlines in 3D vector fields. Presenting an information theoretic framework for flow visualization with a focus on streamline generation are Xu et al. [XLS10]. The authors evaluate the effectiveness of visualizations by measuring how much information in the data is communicated from the visualization.

Flow Topology Theisel et al's approach to constructing saddle connectors in place of separating stream surfaces is an effort to address the challenges of occlusion [TWHS03]. The saddle connectors represent the separation surfaces as a finite number of stream lines. These stream lines are the intersection curves of the separation surfaces, and are called saddle connectors because they start and end in saddle points of the vector field. This is extended by Weinkauf et al. [WTHS04] using separating surfaces and connectors originating from boundary switch curves. Boundary switch curves consist of all points on the boundary where the flow direction is tangential to the boundary surface, and separate outflow and inflow areas.

Peikert et al. [PS09] present topology based methods for constructing stream surfaces in the neighborhood of critical points. The authors discuss error bounds and give application examples for the range of topological features under consideration. Topological methods are of limited use for this application. First, no sources or sinks are present in our 3D flow simulations. Periodic orbits are quite rare and saddle points are certainly not the only features of interest to the user. Plus, explicit extraction of topology also relies on special threshold values which may result in overlooked features of interest [LHZP07]. We demonstrate how our algorithm seeds stream surface curves in areas of interest such as rotational flow, high curvature and saddle points.

Surface Rendering and Visualization Löffelmann et al. [LMGP97] introduce methods for placing arrows on stream surfaces which improve the perception of the flow. Loffelmann et al. [LMG97] then improve on this using hierarchical techniques to optimize the surface tiling. Laramee et al. investigate swirl and tumble with a comparison of visualization techniques [LWSH04]. Laramee et al. [LGSH06] draw on previous image based texture advection research in order to improve the information content and perception of flow on stream surfaces.

Techniques for the illustration of stream surfaces are presented by Born et al. [BWF*10] and Hummel et al. [HGH*10]. Born et al. [BWF*10] describe techniques such as contour lines and halftoning to show the overall surface shape. Flow direction as well as singularities on the stream surface are depicted by illustrative surface streamlines. Hummel et al. [HGH*10] examine algorithms for how transparency and texturing can be used to convey both shape and directional information. These papers concentrate on illustrative renderings of the surfaces while maintaining interactivity.

Surface Seeding Strategies Many papers have been published presenting algorithms for the construction of integral surfaces, the work by Edmunds et al. [EML*11] presents an automatic seeding algorithm for stream surfaces seeded at the boundary of the domain.

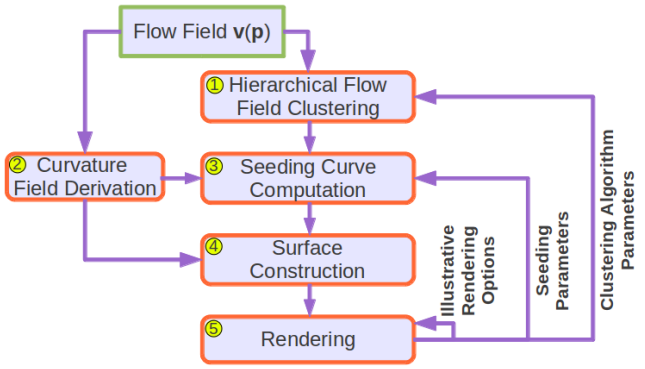

Figure 1: *The automated stream surface seeding pipeline. The pipeline shows the vector field clustered with customization parameters, the derivation of the curvature field, and seeding curve generation. Stream surfaces are then propagated from the seeding curves through the vector field, and then rendered.*

Isolines are derived at the boundary based on exit flow trajectory. These isolines are then used to seed surfaces. The limitation of this approach is that the user may only seed where the flow exits the domain.

3. Automated Stream Surface Seeding

This section describes the stream surface seeding algorithm for the visualization of vector fields. We start with an overview of the processing pipeline illustrated in Figure 1. The algorithm is presented in multiple stages consisting of the clustering process, calculation of the curvature field used to support seeding and illustration, seeding curve generation, stream surface generation, and rendering. The choice of clustering for generating good seed locations is based on the premise that the clustering process can guide seeding, at varying levels of simplification, in the vicinity of the flow features of interest to the user. Examples of this can be seen in section 4. An evaluation can be found in section 5. This process results in interesting stream surfaces which represent the flow structures contained within the domain.

1. Clustering is performed on the flow field with options that guide the clustering process. The parameters provide the user with great flexibility in controlling the density of surfaces and prioritizing the formation of clusters at locations corresponding to sub-sets of the flow interesting to the user. Vector direction, magnitude, and location are comparison attributes that can be set. See Section 3.1.
2. The curvature field is derived from the flow field. It is used to compute seeding curves after the clustering process has identified the seeding locations. It is also used to map opacity to stream surfaces for illustrative rendering. See Section 3.2.
3. The error level parameter, or simplification level s_l, controls which clusters are selected from the hierarchical tree providing the required level of detail or seeding density. Seeding curve generation starts with the selection of representative clusters. The cluster location is automatically used as the basis of the seeding curve. The seeding curves are then generated by integrating forward and backward through the curvature field at a length proportional to the cluster size. This generates seeding curves which follow the local flow structures while maintaining orthogonality with the flow. See Section 3.3.
4. Once the seeding curves are computed, stream surfaces are propagated from each of the seeding curves. Flow attributes such as velocity magnitude may be color mapped. Opacity can be mapped to local flow curvature. See Section 3.4.
5. After the generation of the stream surfaces, the surface data is rendered using a number of illustrative techniques to enhance the perception of the flow field. Opacity derived from curvature of the flow field is utilized in conjunction with depth peeling for rendering semi-transparent surfaces. See Section 3.5.

3.1. Customized Clustering

The algorithm presented in this paper adapts the clustering method of Telea and Van Wijk [TvW99]. The aim of this work is to generate a globally simplified vector representation while maintaining enough detail to represent complex local flow structures. This is achieved using a process which progressively evaluates pairs of neighboring cells according to a distance metric that minimizes error. The error is calculated from two elliptic similarity functions: one compares direction and magnitude, and the second compares location of the neighboring vectors.

The clustering process iterates until a single cluster remains n_{root}. This is the root of a binary hierarchical tree. The clusters at each level of the tree store the resultant vector as a product of the child pair. This results in a tree of size $n_t + n_t - 1$ where n_t is the quantity of initial clusters. The algorithm displays the clusters representative vectors at a user specified level of the hierarchical tree s_l. The accumulated error in the paired clusters is inversely proportional to s_l which is designed to directly correlate with the number of clusters selected from the hierarchical tree and therefore the number of seed locations. See subsection 3.1.3. This combined with the user defined parameters for the error function provides a flexible range of stream surface density and locality which can be tailored to the users requirements.

3.1.1. Clustering Process

The input of the algorithm is a 3D steady state vector field $\mathbf{v}(\mathbf{p}) \in \mathbb{R}^3$ where $\mathbf{v}(\mathbf{p}) = \begin{bmatrix}\mathbf{v}_x(x,y,z) & \mathbf{v}_y(x,y,z) & \mathbf{v}_z(x,y,z)\end{bmatrix}$ for $\mathbf{p} \in \Omega$, $\mathbf{v} \in \mathbb{R}^3$ and $\Omega \subset \mathbb{R}^3$, where Ω is a 3D uniform grid.

The first step requires each $(\mathbf{p}_i, \mathbf{v}(\mathbf{p}_i))$, where $\mathbf{p}_i$ is the grid location, to be stored as a cluster n_i in an initial list of clusters, or cluster set. Each n_i is evaluated against each of its neighbors in Ω. For each neighbor n_j the cluster pair (n_i, n_j) is stored in a hash table along with the derived error ε as the key. The hash table stores all (n_i, n_j) in increasing order of ε.

The clustering process retrieves, in increasing order of ε from the hash table, each (n_i, n_j). If both (n_i, n_j) have not previously been clustered, then (n_i, n_j) are merged. Once merged the new cluster n_k is then compared to each of its neighbors, storing each pair in the hash table in increasing order of ε. This process continues until there are no more pairs to merge. The result is a single cluster covering the entire domain with a single representative vector.

3.1.2. Distance Metric and Cluster Merging

The clustering algorithm utilizes two important functions. The first is the error estimation function, and the second is the merging function. The cluster error function describes the error generated by merging (n_i, n_j) i.e. the similarity between (n_i, n_j). In addition to this we implemented an alternative error function that provides additional flexibility when less similarity is desired. The cluster merging function dictates how the representative vector is calculated from (n_i, n_j).

The elliptic iso-contour of a given error is the locus of the apex of all vectors $\mathbf{w}$ at the same location as $\mathbf{v}$ which are equally similar to $\mathbf{v}$. The elliptic error function defines this relationship. We refer the reader to Telea and Van Wijk [TvW99] for a comprehensive description of the two elliptic error functions $t(\alpha, \beta, \gamma)$ and $s(d,e)$. Their linear combination $l(s,t) = As + (1-A)t$ as a function of $A \in [0,1]$ describes their relationship. $t(\alpha, \beta, \gamma)$ describes the direction and magnitude error function, and $s(d,e)$ describes the positional error. Parameters α, β and γ are set by the user, and d, e are set by specifying $B \in [0,1]$ where $B = d/e$ and $d + e = 1$.

Elliptic Error Functions Our experiments with the elliptic error functions produced interesting results when used for stream surface seeding. However we observe limitations which are addressed with a customized distance function.

When modifying the input parameters, a range of characteristics are observed. For example, when $A = 0.9$ emphasizing position,

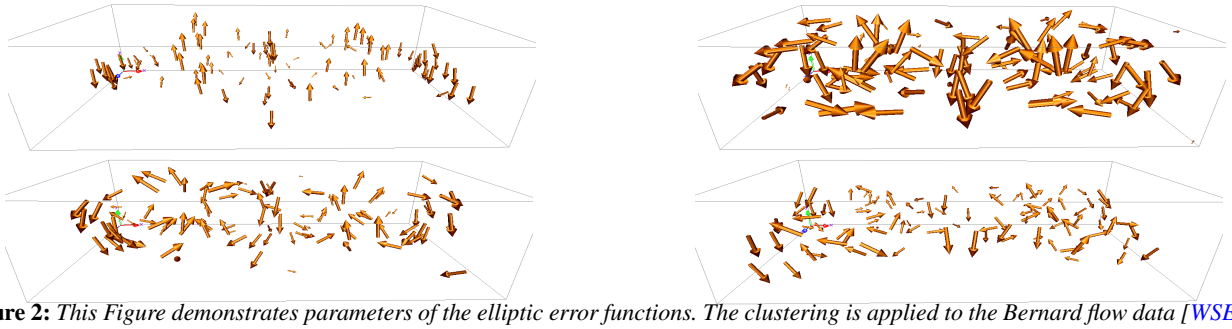

Figure 2: *This Figure demonstrates parameters of the elliptic error functions. The clustering is applied to the Bernard flow data [WSE05]. The glyphs visualize the representative vector at the cluster location. The parameters of the top left image are $A = 0.1$, $B = 0.25$ and $s_l = 100$. The glyphs are fairly evenly distributed. The parameters of the top right image are $A = 0.1$, $B = 0.75$ and $s_l = 100$. The glyphs are fairly evenly distributed around the rotating areas of flow. The parameters of the bottom left image are $A = 0.9$, $B = 0.25$ and $s_l = 100$. The glyphs are concentrated around the rotating areas of flow. The parameters of the bottom right image are $A = 0.9$, $B = 0.75$ and $s_l = 100$. The glyphs are fairly evenly distributed.*

and $B = 0.75$ emphasizing neighbors along the vector direction, the process produces an evenly distributed set of clusters at a given s_l. See Figure 2 bottom right. Setting $A = 0.9$ emphasizing position and $B = 0.25$ emphasizing neighbors orthogonal to the vector direction, concentrates the clusters around areas of curved flow. See Figure 2 bottom left. In these images it can be observed that to visually represent the flow a large number of locations are required. This would lead to a very cluttered incoherent visualization when seeding surfaces at all the locations.

Our aim is to generate seeding curves in optimal locations adjacent to the flow structures with minimal density i.e. enough to represent the flow structure without clutter. Seeding curves at the center of highly curved structures such as vortices is not ideal and may result in complex surfaces that are difficult to discern. The complexities include high curvature, shearing, crossing, and diverging surfaces.

The ideal solution is to construct the minimum quantity of seeding curves at locations of interesting flow and at a non zero distance from the centers of complex, curved structures. Flow structures are characterized by their curvature and velocity gradient. The vortex is one example of this. See Figure 3.

Generating clusters to emphasize pairing in areas where there is a non zero velocity gradient motivates us to define a parameter which supports this. For example if the error function returned lower ε for cluster pairs which have a greater differences in magnitude than their neighbors, while favoring orthogonal clusters e.g. $B = 0.25$, we can predict a concentration of clusters in areas of greater velocity gradient. Extending this strategy to vector direction, we can specify a concentration of clustering in areas of high curvature while emphasizing clusters orthogonal to the vector direction e.g. $B = 0.25$.

In the remainder of this section we present our customized distance function which is designed to implicitly capture, in combination, areas of non zero velocity gradient and areas of curved or rotating flow. In principle we are pattern matching attributes of the compared vectors using unsupervised machine learning or agglomerative hierarchical (binary) clustering. To achieve this our parameters are constructed to return low error not just for similar, but also dissimilar directions or magnitudes of a quantified amount.

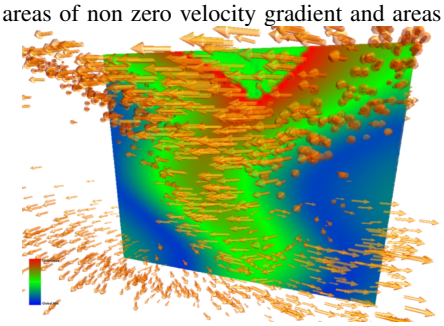

Figure 3: *A 2D slice taken at the center of the Tornado simulation. Velocity is mapped to color where blue is minimum velocity and red is max. This illustrates the velocity gradient initially increasing and then decreasing away from the center of the vortex.*

A Customized Distance Function We start with decoupling the direction $\hat{\mathbf{v}}$ and magnitude $|\mathbf{v}|$ components of the distance function. We then combine them with the position function in a manner which enables emphasizing a particular direction or difference in magnitude rather than similarity. This is effectively pattern matching, rather than assessing similarity.

To combine the direction ε_δ, magnitude ε_μ, and position ε_ψ errors we simply linearly combine them. The user options for specifying the emphasis of a particular direction $\eta_\delta \in [0,1]$, difference in magnitude $\eta_\mu \in [0,1]$ or position $\eta_\psi \in [0,1]$ are included:

$$\varepsilon = \varepsilon_\delta(\eta_\delta) + \varepsilon_\mu(\eta_\mu) + \varepsilon_\psi(\eta_\psi)$$

where $\varepsilon \in [0,1]$.

Direction The user controlled coefficient η_δ corresponds to parallelism or orthogonality. Setting $\eta_\delta = 0$ emphasizes parallel vectors $\mathbf{v} \parallel \mathbf{w}$, whereas $\eta_\delta = 1$ emphasizes opposing vectors. $\eta_\delta = 0.5$ emphasizes orthogonal vectors $\mathbf{v} \perp \mathbf{w}$. The linear combination is specified as:

$$\varepsilon_\delta(\eta_\delta) = (\eta_\delta \cdot \delta) + (1 - \eta_\delta) \cdot (1 - \delta)$$

Where $\varepsilon_\delta \in [0,1]$ represents the direction centered error, with lower values favored by the clustering process, and $\delta \in [0,1]$ is:

$$\delta = \frac{(\hat{\mathbf{w}} \cdot \hat{\mathbf{v}}) + 1}{2}$$

Magnitude Setting $\eta_\mu = 0$ emphasizes vectors of equal length $|\mathbf{v}| = |\mathbf{w}|$, whereas $\eta_\mu = 1$ emphasizes vectors of different length $|\mathbf{v}| \neq |\mathbf{w}|$. The linear combination is:

$$\varepsilon_\mu(\eta_\mu) = (\eta_\mu \cdot \mu) + (1 - \eta_\mu) \cdot (1 - \mu)$$

Where $\varepsilon_\mu \in [0,1]$ represents the magnitude centered error, with lower values favored by the clustering process, and $\mu \in [0,1]$ is the absolute value of μ' defined as:

$$\mu' = |\mathbf{v}|/(|\mathbf{v}| + |\mathbf{w}|) - |\mathbf{w}|/(|\mathbf{v}| + |\mathbf{w}|)$$

Position For position we use the elliptic error function by Telea and Van Wijk [TvW99]. For consistency our user controlled coefficient η_ψ corresponds to B, and $\varepsilon_\psi(\eta_\psi)$ corresponds to s where $\varepsilon_\psi \in [0,1]$.

Default Settings From experimentation with these customized distance function parameters we have derived two sets of values which provide the user with either a feature-centered or overview setting. For the feature-centered setting $\eta_\psi = 0.25$, $\eta_\delta = 0.3$, $\eta_\mu = 0.2$. For

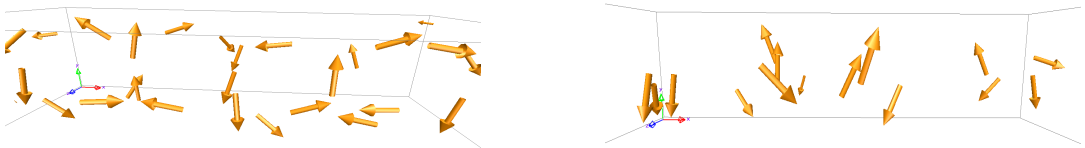

Figure 4: *The left image is the Bernard flow simulation visualized with* $\eta_\psi = 0.25$, $\eta_\delta = 0.0$ $\eta_\mu = 0.5$ *and* s_l *= 27. The glyphs demonstrate clustering in the vicinity of the velocity gradient associated with the vortices. The right image demonstrates cluster representations focusing around cores of the vortices. This is achieved using parameters which emphasize clustering of vectors with orthogonal directions.* $s_l = 27$, $\eta_\psi = 0.5$, $\eta_\delta = 0.5$ *and* $\eta_\mu = 0.0$.

the overview settings $\eta_\psi = 0.75$, $\eta_\delta = 0.0$, $\eta_\mu = 0.0$. From hereafter we will refer to these settings as feature-centered or overview. Further discussions of clustering parameters reside in section 4.

The Merging Function The union of a pair of clusters (n_i, n_j) results in a parent cluster n_k. The new n_k stores a representative vector $\mathbf{v}(n)$ and location $\mathbf{p}(n)$. This is achieved by computing a volume weighted average of (n_i, n_j) representative vectors and locations. Each level of the tree contains clusters of increasing ε.

3.1.3. Simplification Level

The simplification level s_l is an option for the user to select clusters or seeding locations from the cluster hierarchy at rendering time. Error metric ε is used in the first step of the clustering algorithm to place pairs (n_i, n_j) in the hash table in order of increasing ε. A second step clusters them in this order marking each new cluster with a unique order number l incremented from 0. As a result ε is directly related to l. Once the tree has been built this order number l is compared to the user parameter s_l to determine cluster selection for seeding or display. The following describes the computational relationship of s_l, l and ε and how the clusters are selected from the tree. Because we use a binary tree the root cluster's level $l(n_{root}) = n_t - 1$ where n_t is the quantity of initial clusters.

To select clusters from the tree, the user specifies s_l in the range $s_l \in [1, n_t - 1]$. We next calculate l where $l = n_t - s_l$. The level l is then used to search the tree for all clusters with levels $l(n) \leq l$ with parents having $l(n) > l$. The set of clusters which meet this condition are returned as the seeding locations. s_l is designed to be equal to the number of clusters or seeding locations, and is inversely proportional to ε. s_l provides consistency for the user across different simulation data for selecting the quantity of seed locations. See Figure 4.

3.2. Curvature Field Derivation

From the vector field $\mathbf{v}(\mathbf{p})$ we derive the curvature field Ω_c. This step allows the curvature data to be sampled or interpolated on demand. The role of Ω_c is to support seeding curve computation (Section 3.3) and illustrative techniques for rendering the stream surfaces (Section 3.5). Ω_c is not needed for the clustering stage of the pipeline.

Each curvature sample $\mathbf{c}(\mathbf{p}) \in \Omega_c$ defined for $\mathbf{p} \in \mathbb{R}^3$ and $\Omega_c \subset \mathbb{R}^3$, is derived by applying a combination of operators to the vector field. Ω_c is derived from the first derivative $\mathbf{v}$ and second derivative $\mathbf{a}$ of the flow field. The second derivative $\mathbf{a}$ is acceleration and defined as $\mathbf{a} = (\nabla\mathbf{v})\mathbf{v}$ where $\nabla\mathbf{v}$ is the Jacobian of the velocity field. Steady state curvature [Rot00] is defined as:

$$\mathbf{c} = \frac{\mathbf{v} \times \mathbf{a}}{|\mathbf{v}|^3}$$

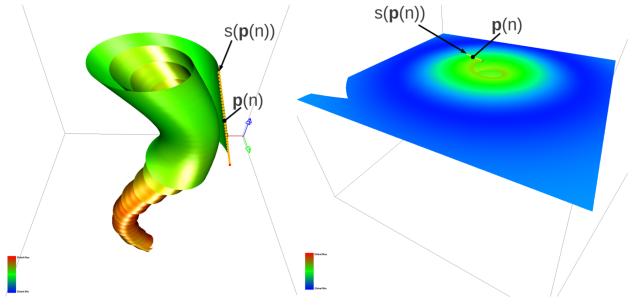

Figure 5: *Rotating flow is more intuitively represented in the left figure. The left figure uses a surface tangent to its curvature and parallel with the axis of rotation. The right figure is represented by a surface perpendicular to the axis of rotation and flow curvature.*

3.3. Seeding Curve Computation

The seeding curve generation commences following the clustering. Every n_i in the selected representation stores information regarding its location $\mathbf{p}(n)$, its level $l(n)$, its error $\varepsilon(n)$, its representative vector $\mathbf{v}(n)$ and size $vol(n)$. $\mathbf{p}(n)$ defines the center of each new seeding curve $s(\mathbf{p}(n))$. Location alone is not enough to generate effective seeding curves. The locations given by the cluster representation provide interesting seeding positions that capture the local characteristics of the flow. To take advantage of the potential seeding locations we ensure the seeding curves are positioned, orientated and scaled according to the local flow characteristics. The position is given by the cluster. The orientation and size of the seeding curves require analysis.

3.3.1. Seeding Curve Orientation

The orientation of the seeding curve stems from the observation that seeding curves generally produce informative surfaces maximizing coverage when orthogonal to the flow.

In addition the seeding curve should be oriented with respect to the local flow structures. If this point is ignored the surface can impart a less useful visualization to the viewer. For example if a surface is tangent to the local curvature and parallel to the axis of rotation it intuitively represents rotating flow. If the flow is represented by a surface perpendicular to the axis of rotation and flow curvature, the representation is less informative. See Figure 5. We position a straight seeding curve $s(\mathbf{p}(n))$, centered at $\mathbf{p}(n)$, orientated in line with the local curvature vector $\mathbf{c}(\mathbf{p}(n))$. The curvature vector $\mathbf{c}(\mathbf{p}(n))$ is sampled from the curvature field Ω_c at $\mathbf{p}(n)$.

The result of this approach is a seeding curve which is orthogonal to the local flow at its center. The seeding curve is also orientated such that it is parallel to the axis of local flow rotation. In Figure 6 we see examples of the proposed seeding curve orientation. The Figure shows how the seeding curve orientation relates to the flow characteristics, and the potential to provide intuitive visualizations of the nearby flow structures.

3.3.2. Seeding Curve Length

The length of a seeding curve $s(\mathbf{p}(n_i))$ is proportional to the volume of cluster n_i. The volume of cluster n_i is equivalent to the volume of zero level (or initial) clusters contained within n_i. Specifying the length of the seeding curve as equivalent to the clusters volume may lead to the curve exiting the boundaries of Ω. To refine this we use the cubed root of the cluster volume:

$$l(s) = \sqrt[3]{vol(n_i)}$$

This is based on the premise that a single cluster representing the whole domain would require a seeding curve which extends to its boundaries. This approach works well for all the data we tested.

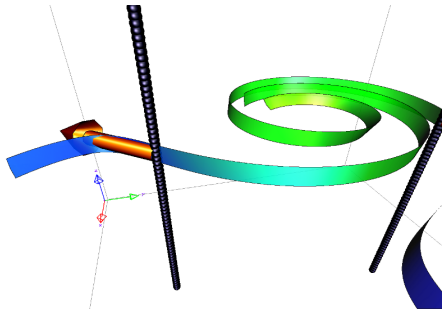

Figure 6: *The Tornado simulation illustrates the alignment of seeding curves to the curvature field. The feature-centered values are specified and $s_l = 2$. The arrow glyph visualizes the representative vector and location. The seeding curves (black) orientation to flow curvature is emphasized using stream ribbons.*

However, the user is also provided with an optional scaling coefficient for further customization. This enables shorter seeding curves when seeding with a denser cluster representations, and vice versa. Figure 7 illustrates different seeding curve lengths proportional to cluster density.

3.3.3. Seeding Curve Integration

Our Experiments with this method produce interesting results. However it is clear that further refinement of the seeding curve is desired. In areas of flow which demonstrate high curvature seeding curves may cross each other. See Figure 8. Longer seeding curves can present problems. As the distance away from the seeding curve center increases, its orientation relative to the local flow direction can be non optimal. For example the seeding curve is orthogonal to the flow at its center, but may be in line with the flow at its extremities. Additionally the seeding curves do not follow the local flow structures we aim to capture with the clustering process. For example a straight seeding curve generated neighboring a vortex curved along its core line would either extend away from or cross the vortex. This could produce non optimal representations of the flow structure.

To address these challenges we further enhance the seeding curve generation. A logical approach is to exploit the curvature field Ω_c. The curvature vector is the cross product of velocity and acceleration, and therefore always orthogonal to velocity. This is an ideal candidate as we generate a seeding curve orthogonal to the flow along its the complete length while following the local curvature of the flow field.

Starting at $\mathbf{p}(n)$ the seeding curve $s(\Omega_c(\mathbf{p}(n)))$ is generated by integrating through the curvature field $\Omega_c(\mathbf{p}(n))$. The integration step is defined by the required seeding curve discretization. See section 3.4. The length of the curve is constrained as previously described. The integration is achieved using a fourth order Runge Kutta integrator, sampling the curvature field $\Omega_c(\mathbf{p}(n))$, in forward and backward directions. Examples of this method can be seen in Figure 9. It is possible within the curvature field to find degenerate points if $\mathbf{v}$ and $\mathbf{a}$ are equal. We handle the degenerate points by simply terminating the integration.

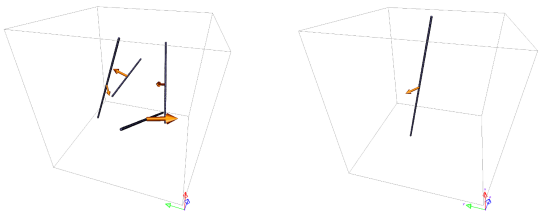

Figure 7: *These figures show that the seeding curve length is proportional to the volume weighted average of the clusters children. The left figure is represented by 4 clusters and the right figure by 1.*

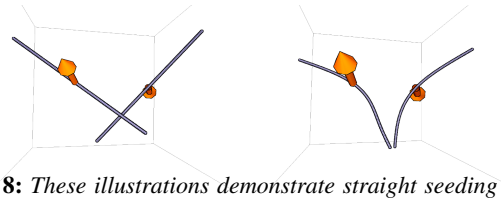

Figure 8: *These illustrations demonstrate straight seeding curves vs integrated seeding curves. The representative vectors and locations are visualized with arrow glyphs. The left image shows the seeding curves crossing each other when aligned with the curvature vector. The right image shows the seeding curves following the curvature, not crossing, and remain orthogonal to the flow.*

3.4. Stream Surface Generation

Our work utilizes an out of the box solution for generating stream surfaces. The stream surface is constructed with an advancing front scheme first introduced by [Hul92]. An adaptive Runge Kutta 4th order integrator is used in the surface construction sampling the data with a tri-cubic interpolation method [LM05]. We allow the user to select downstream or/and upstream stream propagation e.g. forward and reverse integration. By default surfaces are terminated when they leave the domain, enter a periodic orbit, or reach a predetermined maximum length. The user has an option to control the length.

During the construction of the stream surface it is useful to sample attributes of the flow which can be stored with the vertex data representing the surface. This data is then be passed to the rendering pipeline for further processing as described in 3.5. Attributes of use can include color mapping $|\mathbf{v}|$. The alpha channel can be mapped to $|\mathbf{c}|$. This is particularly useful when rendering semi-transparent surfaces with depth peeling as described in 3.5.

3.5. Rendering Enhancements

A number of techniques are implemented to aid the viewer in perception of the resulting visualization. The techniques include the use of transparency, color, silhouette edge highlighting, lighting and shading. We incorporate semi-transparency with a combined approach. First as part of the stream surface construction algorithm we can assign alpha values from Ω_c. For example with higher curvature $|\mathbf{c}|$ we assign a higher alpha value α_c. Note: $max(|\mathbf{c}|)$ is the maximum curvature in Ω_c, and is used to normalize $\alpha_c \in [0,1]$. This has the effect of increasing opacity of the inner structures of curved surfaces:

$$\alpha_c = \frac{|\mathbf{c}|}{max(|\mathbf{c}|)}$$

The second part of our combined approach is a view dependent strategy which uses the relationship α_v between the view normal $\hat{\mathbf{n}}_\mathbf{v}$ and the surface normal $\hat{\mathbf{n}}_\mathbf{s}$ defined as:

$$\alpha_v = \frac{2cos^{-1}(\hat{\mathbf{n}}_\mathbf{s} \cdot \hat{\mathbf{n}}_\mathbf{v})}{\pi}$$

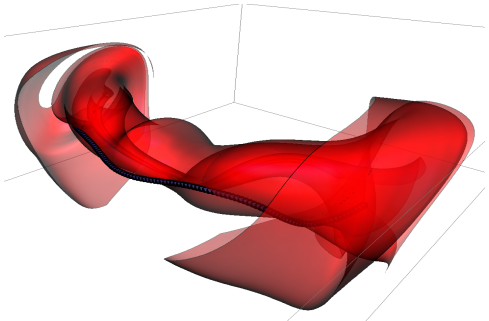

Figure 9: *This illustration of the Bernard simulation demonstrates the seeding curve following the flow structure. The seeding location derived from the clustering is at the center of the curve (black).*

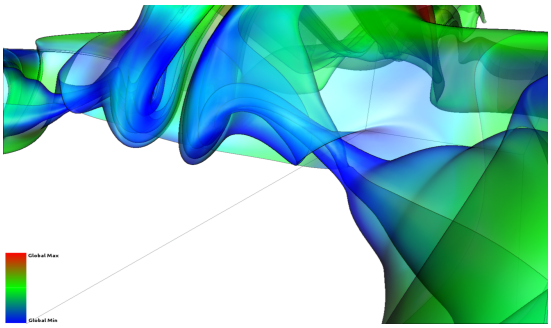

Figure 10: *The flow past a cuboid simulation demonstrating illustrative techniques on a stream surface generated neighboring a double vortex structure emanating from a critical point.*

where $cos^{-1}(\hat{\mathbf{n}}_\mathbf{s} \cdot \hat{\mathbf{n}}_\mathbf{v})$ is given in radians. This angle based relationship has the effect of increasing the transparency when the surface tends to face the view port, or reducing transparency when the surface is orthogonal to the view port. A linear relationship with a user specified bias η_α is used to combine these aspects:

$$\alpha = \eta_\alpha \alpha_c + (1 - \eta_\alpha)\alpha_v$$

The application of transparency to visualizations poses problems relating to the order of primitive rendering. To solve this issue we use depth peeling, an order independent transparency technique presented by Bavoli and Myers [BM08].

Silhouette edge highlighting is used to help the viewer in perceiving where the surfaces curve away from the viewer, and enhance surface edges. Silhouette highlighting utilizes a simple Gaussian kernel in image space [MH02]. The technique is applied to a depth image from the current render pass and is blended to the frame buffer. This way it can be used during normal scene rendering or interleaved with the depth peeling loop.

In addition, reducing the saturation of color as the surfaces curve away from the viewer further enhances the perception of shape. To implement this we modify the magnitude of the *RGB* values during the rendering process. An efficient method is to reuse α_v. This angle based relationship is then used to scale the *RGB* values RGB_{in} sent to the pipeline: $RGB = RGB_{in}(1 - \eta_{RGB}\alpha_v)$ where η_{RGB} is a user supplied bias. A suitable value for η_{RGB} and η_α derived from experimentation is 0.5. However further refinement is made available to users if they so desire. Examples of these settings are shown in Figure 10.

4. Results

The key to an informative visualization is the ability to capture the flow characteristics and to represent different data attributes. We discuss the results and the performance of our algorithm. The results demonstrate the algorithm applied to a range of data, both analytical and computational. The different simulations provide a diverse environment testing the robustness of the algorithm. We demonstrate the clustering, seeding, and illustration options on each of the simulations.

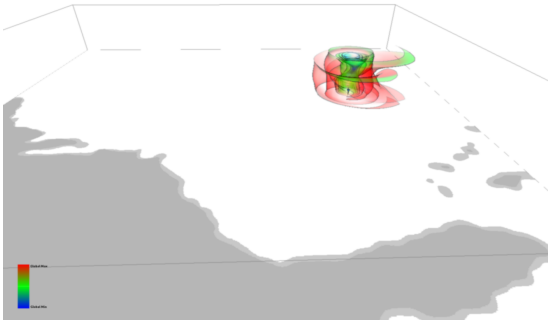

Figure 11: *Hurricane Isabel data visualized with automatic stream surfaces. Color is mapped to velocity, and opacity is mapped to vector field curvature. This visualization emphasizes the eye of the hurricane captured by stream surfaces rendered with edge highlighting and view dependent color saturation. The inner structure of the vortex tends away from blue as the velocity increases away from the center to the vortex.*

Hurricane Isabel This is a simulation from the IEEE Visualization Contest 2004. The hurricane modeled is Hurricane Isabel which occurred in September of 2003. The clustering and surface seeding process is applied to the Isabel flow data with different sets of parameters designed to show the flexibility of the algorithm.

The first set of parameters are $\eta_\psi = 0.25$, $\eta_\delta = 0.3$, $\eta_\mu = 0.2$ (feature-centered), with $s_l = 3$. This produces clusters in the region of the hurricane vortex. Figure 11 shows the stream surfaces which capture the vortex. The visualization uses transparency and edge highlighting to enhance the perception of the rotating flow, and capture the inner structure of the vortex. The use of a higher s_l value produces more seeding locations in this area, moving outwards as the quantity increases. this may be useful in some cases but is likely to become cluttered.

The second set of parameters are designed to give a more evenly distributed set of seeding locations. The parameters are $\eta_\psi = 0.75$, $\eta_\delta = 0.0$, $\eta_\mu = 0.0$ (overview), with $s_l = 30$.

In Figure 12 an overview of the simulation can be seen. The illustrative techniques support the perception of the flow characteristics with transparency enabling the user to understand the inner flow structure of the vortex. It can be seen from this visualization how the general flow behavior interacts with the eye of the hurricane. A second vortex like structure can be seen in the mid left of Figure 12, located above the coast.

Flow Past A Cuboid This simulation demonstrates flow past a Cuboid over 102 time steps. For our experiments we use the last time step containing fully formed flow structures. The clustering process is applied with one set of parameters at different simplification levels. We use $\eta_\psi = 0.25$, $\eta_\delta = 0.3$, $\eta_\mu = 0.2$ (feature-centered), with $s_l = 2$, and $s_l = 4$ respectively.

Figure 13 demonstrates the seeding algorithm at $s_l = 2$. The double vortex structure is clearly visible. Figure 14 illustrates the ability of the user to change to the level of simplification to $s_l = 4$, thus generating a denser visualization by combining the two sets of surfaces providing better contextual information. The visualization captures the prominent characteristics of the vortex shedding behind the cuboid.

Bernard Flow The Bernard flow data is a numerical simulation defined on a regular grid [WSE05]. The simulation demonstrates thermal motion as a result of convection. The clustering process is

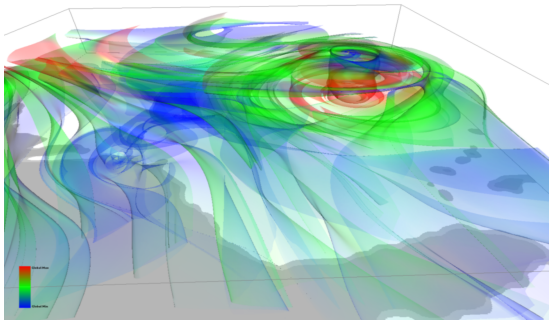

Figure 12: *Hurricane Isabel visualized using settings designed to give a broader representation. A second smaller vortex like structure can be seen on the coast in the mid left of this still image. Stream surfaces are rendered with edge highlighting and view dependent color saturation. The inner structures of the simulation are viewed with additional transparency.*

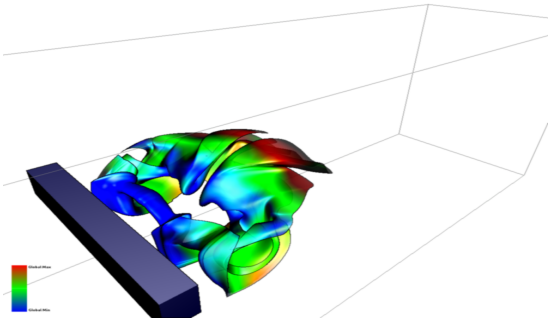

Figure 13: *This visualization of the flow past a cuboid is performed with $s_l = 2$. The seeding curve follows a neighboring double vortex structure emanating from a saddle point. The stream surface is integrated down stream, and is rendered using illustrative techniques.*

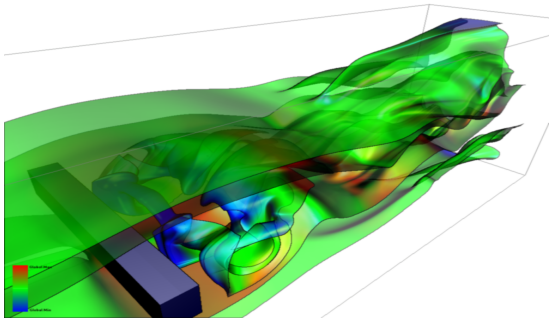

Figure 14: *This image highlights the ability of our algorithm to provide both overview and feature-centered within the same visualization. The vortex shedding is represented by the stream surfaces along with its relation to the rest of the flow field.*

applied to the Bernard flow data with $\eta_\psi = 0.25$, $\eta_\delta = 0.3$, $\eta_\mu = 0.2$ (feature-centered). Once the clustering is complete the user now has the option to specify the level of simplification s_l. A slider is used to adjust the level of detail with visual feedback in the form of glyphs rendered at the cluster locations. For this visualization we specify $s_l = 4$.

Figure 15 shows the final surfaces rendered with transparency. The coefficient $\eta_\alpha = 0.5$ for this visualization demonstrates the usefulness of opacity mapped to Ω_c. Edge highlighting is used to emphasize the surface boundaries with the surface color darkened as the surface curves away from the view port. The curved surfaces and vortex cores are clearly emphasized using these techniques. This visualization is further enhanced with the use of color mapped to $|\mathbf{v}|$. The unit vectors in this numerical data give a constant color across the surfaces, except in areas of degenerate velocity. This is useful for highlighting critical points across the domain. Figure 16 highlights these saddle points at the center and ends of the four double vortices.

Parameters While the selection of a simplification level is simple, as it directly correlates to the number of seeding locations, its selection is important as the final visualization can easily become over populated or too sparse. The selection of parameters is important for producing the type of visualization required by the user as they are, although progressive, sensitive to change. The two proposed default settings are demonstrated in this section. It was found from experimentation that careful thought and a general understanding of fluid flow is required to design sets of parameters that are useful to the flow engineer. A poor choice of parameters could lead to cluttered visualizations of little meaning to an engineer, similar to a naive approach (See Figure 17). These ideas are further evaluated in section 5.

Performance Our algorithm is implemented in C++ and QT4 on a PC with an NVIDIA GeForce GTX480, an Intel quad core 2.8GHz CPU with 8GB RAM. The bottleneck in the performance of our algorithm is the clustering as seen in Table 1. The clustering process compares closely with previous vector field clustering algorithms that operate on uniform grids [PGL*12].

Clustering Performance		
Data	No. of Clusters	Time [ms]
Hurricane Isabel	49,999,999	542,452
Bernard Flow	524,287	2,908
Cuboid	1,179,647	6,898
Lorenz Attractor	4,194,303	34,812

Table 1: *Clustering performance of a range of simulations.*

The field derivation ranges from 8ms for 262,144 data samples to 958ms for 25,000,000 data samples. The seeding curve generation is a function of s_l and curve length which decreases as s_l increases. Timings are from 15ms for $s_l = 30$ to 35ms for $s_l = 100$. Stream surface rendering performance is a function of surface size and complexity e.g. the number of vertices and depth peeling layers. We achieve 20fps+ when rendering meshes of 400k vertices while utilizing 10 depth layers. This is comparable to the work by Born et al. [BWF*10] and Hummel et al. [HGH*10] who discuss surface rendering performance in detail.

5. Domain Expert Evaluation

To evaluate the usefulness of the proposed framework in the visualization of complex CFD data, we have shown our visualization results and system to a number of CFD experts, i.e., Dr. Malki, Dr. Masters, and Dr. Croft, co-authors of this paper. This section summarizes their positive and constructive feedback.

One of the most commonly used software packages for the visualization of numerical simulation results is Tecplot, which does not offer the user the option to create stream surfaces (only stream tubes). An example is a vertical axis wind turbine [MMWC11], where the classical analysis partitions the flow into vertical slices. Both streamlines and stream surfaces have their uses, however, where 2D images are extracted for use in reports or presentations and the user cannot rotate the model to evaluate it from different viewing angles, streamlines can be quite messy and difficult to comprehend. Stream surfaces on the other hand are much neater and clearer showing how different layers of flow may fold over each other.

Surfaces are very helpful in defining boundaries in the flow. The particular approach of this paper, where the surface seeding line is curved in response to the physical data, gives a surface that is analogous to the classical definition of the boundary of a stream

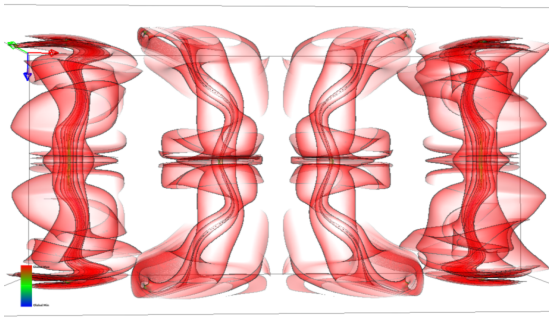

Figure 15: *The Bernard Flow numerical simulation visualized using our algorithm. The seeding locations are derived from the clustering process using the feature-centered parameters and $s_l = 4$. The four main vortex structures are clearly emphasized. The thermal motion of the flow field is captured with our framework. The figure shows the surfaces rendered with transparency mapped to α_c and α_v.*

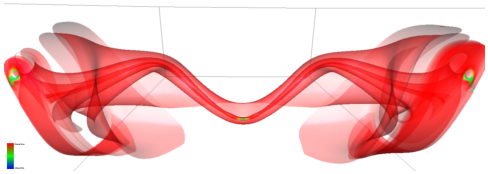

Figure 16: *The Bernard Flow numerical simulation visualized with $s_l = 4$, and the feature-centered parameters. This figure shows one of the 4 surfaces which highlights saddle points within the domain. The stream surface is rendered red, while the degenerate areas are highlighted green to blue.*

tube [DGS79] or stream slice [Par82] [Hom91]. This gives the surface two properties that are very useful. First it is a solid surface, partitioning the flow into regions, and secondly the surface relates (approximately) to the boundaries of stream tubes, when you realize that a stream tube can be any arbitrary shape [Hom91].

Compared to manual seeding, there is a clear advantage of using automated methods as they are able to identify regions where interesting features occur within the flow which may otherwise be overlooked. This is particularly relevant where the user does not have a clear understanding of the flow structure through the domain and hence, may not necessarily be able to identify the locations of such features to select the starting points of stream tubes or surfaces.

The experts from the CFD community welcome the flexibility of control of seeding parameters, although it may be sufficient to provide the user with a few standard parameter sets to quickly provide the user with alternative views of the data. However, there are likely to be situations where the user is particularly interested in a specific region, even if it is not necessarily the most exciting location in terms of flow structure. It would be useful in such situations for the user to have more control over what is shown, in what level of detail and where within the domain.

Varying the number of clusters enables the user to evaluate the flow structure in different levels of detail. Fewer clusters can be used to evaluate the most complex regions of the flow, such as the swirling region of a turbine wake [MMWC11], whilst the number of clusters can be significantly increased to evaluate the far-field regions and obtain a more general picture of the flow throughout the entire domain. Together, these can be used to build a more complete picture of how the flow is behaving throughout the domain.

Applying transparency to the stream surfaces enables the user to view much more detail in terms what is happening within the flow at different levels and understand how the flow structure is developing. Also, it is very useful to use a color scale showing the variation on various parameters e.g., velocity, pressure or turbulence intensity. This provides much more information to the user in the process of trying to identify and better understand the flow characteristics.

This framework is useful for the identification and visualization of interesting flow features such as swirl, and for visualizing the interaction of wakes behind obstructions with downstream objects

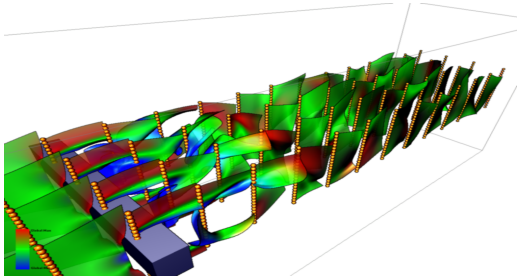

Figure 17: *A naive seeding approach to the flow past a cuboid simulation. Although the range of illustrative techniques are applied the perception of the flow characteristics is limited. The surfaces are seeded at regular intervals at a consistent orientation across the domain.*

falling within the wake's flow path. The tool would be of particular use when visualizing the results for the benefit of someone who may not be entirely familiar with the details or nature of the data.

6. Conclusions

This paper introduces a novel automated stream surface seeding strategy based on clustering. The algorithm automates the capturing and visualization of important characteristics within the flow field as defined by the user. The user is able to guide the visualizations by specifying the feature-centered or overview clustering parameters. Emphasis is placed on flexibility allowing the density of seeding curves and their associated stream surfaces to be controlled by the user. The novel adaption of an existing clustering method provides greater flexibility over the formation of the clusters and their locations. This approach leads to automatically locating seeding positions near important structures within the domain.

The novel technique for orienting straight seeding curves in line with the local curvature, is further enhanced utilizing the characteristics of the curvature field. The seeding curves are generated by integrated through the curvature field thus following the curvature of the flow structures. Illustrative techniques providing enhancements to the perception of the visualization are used.

The goal of this work is to introduce an algorithm with which we can design and implement stream surface placement. We demonstrated two examples of this with the feature-centered and overview default settings. The domain experts concluded that a set of standard parameter combinations would be beneficial for normal use. As a result of the flexibility of this framework, standard settings can be designed for use according to the requirements of the engineers using our technique.

Since this is the first complete stream surface placement algorithm comparisons with competing work is not possible. However, this work is evaluated by experts within the CFD community who conclude that this technique produces more informative results with less difficulty than with a manual approach. Future work includes an examination of the memory requirements associated with extending this algorithm to time dependent flow and very large simulations. While not the goal of this work accuracy and speed are also areas which will require further study.

7. Acknowledgments

The Authors would like to thank the Department of Computer Science at Swansea University, UK, the Department of Computer Science at the University of Utah, US, and the Department of Computer Science at Oregon State University, US.

Guoning Chen was supported by DOE SciDAC VACET. This work was undertaken in collaboration with the Low Carbon Research Institute Marine Consortium (www.lcrimarine.org).

References

[BM08] BAVOLI L., MYERS K.: Order Independent Transparency with Dual Depth Peeling. *NVIDIA Developer SDK 10* (February 2008). 7

[BWF*10] BORN S., WIEBEL A., FRIEDRICH J., SCHEUERMANN G., BARTZ D.: Illustrative Stream Surfaces. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1329 –1338. 1, 2, 8

[CCK07] CHEN Y., COHEN J. D., KROLIK J.: Similarity-Guided Streamline Placement with Error Evaluation. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1448–1455. 2

[DGS79] DOUGLAS J. F., GASIOREK J. M., SWAFFIELD J. A.: *Fluid Mechanics*, 1 ed. Pitman Publishing Limited, 1979. 9

[DH93] DELMARCELLE T., HESSELINK L.: Visualizing Second-order Tensor Fields with Hyperstream lines. *IEEE Computer Graphics and Applications 13*, 4 (July 1993), 25–33. 2

[EML*11] EDMUNDS M., MCLOUGHLIN T., LARAMEE R. S., CHEN G., ZHANG E., MAX N.: Automatic Stream Surfaces Seeding. In *EUROGRAPHICS 2011 Short Papers* (Llandudno, Wales, UK, April 11–15 2011), pp. 53–56. 2

[FCL09] FORSBERG A., CHEN J., LAIDLAW D.: Comparing 3D Vector Field Visualization Methods: A User Study. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1219–1226. 2

[GKT*08] GARTH C., KRISHNAN H., TRICOCHE X., TRICOCHE T., JOY K. I.: Generation of Accurate Integral Surfaces in Time-Dependent Vector Fields. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1404–1411. 1

[GTS*04] GARTH C., TRICOCHE X., SALZBRUNN T., BOBACH T., SCHEUERMANN G.: Surface Techniques for Vortex Visualization. In *Joint Eurographics - IEEE TCVG Symposium on Visualization* (2004), Deussen O., Hansen C., Keim D., Saupe D., (Eds.), Eurographics Association, pp. 155–164. 1

[HGH*10] HUMMEL M., GARTH C., HAMANN B., HAGEN H., JOY K.: IRIS: Illustrative Rendering for Integral Surfaces. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1319 –1328. 1, 2, 8

[Hom91] HOMICZ G. F.: *Numerical Simulation of VAWT Stochastic Aerodynamic Loads Produced by Atmospheric Turbulence: VAWT-SAL Code*. Sandia National Laboratories, 1991. 9

[Hul92] HULTQUIST J. P. M.: Constructing Stream Surfaces in Steady 3D Vector Fields. In *Proceedings IEEE Visualization '92* (1992), pp. 171–178. 1, 6

[JL97] JOBARD B., LEFER W.: Creating Evenly–Spaced Streamlines of Arbitrary Density. In *Proceedings of the Eurographics Workshop on Visualization in Scientific Computing '97* (1997), vol. 7, pp. 45–55. 2

[Lar02] LARAMEE R.: Interactive 3D Flow Visualization Using a Streamrunner. In *CHI 2002, Conference on Human Factors in Computing Systems, Extended Abstracts* (April 20–25 2002), ACM SIGCHI, ACM Press, pp. 804–805. 2

[LGD*05] LARAMEE R. S., GARTH C., DOLEISCH H., SCHNEIDER J., HAUSER H., HAGEN H.: Visual Analysis and Exploration of Fluid Flow in a Cooling Jacket. In *Proceedings IEEE Visualization 2005* (2005), pp. 623–630. 1, 2

[LGSH06] LARAMEE R. S., GARTH C., SCHNEIDER J., HAUSER H.: Texture-Advection on Stream Surfaces: A Novel Hybrid Visualization Applied to CFD Results. In *Data Visualization, The Joint Eurographics-IEEE VGTC Symposium on Visualization (EuroVis 2006)* (2006), Eurographics Association, pp. 155–162,368. 2

[LHZP07] LARAMEE R., HAUSER H., ZHAO L., POST F. H.: Topology-Based Flow Visualization: The State of the Art. In *Topology-Based Methods in Visualization (Proceedings of Topo-in-Vis 2005)* (2007), Mathematics and Visualization, Springer, pp. 1–19. 1, 2

[LM05] LEKIEN F., MARSDEN J.: Tricubic Interpolation in Three Dimensions. *Journal of Numerical Methods and Engineering 63* (2005), 455–471. 6

[LMG97] LOFFELMANN H., MROZ L., GROLLER E.: *Hierarchical Streamarrows for the Visualization of Dynamical Systems*. Technical report, Institute of Computer Graphics, Vienna University of Technology, 1997. 2

[LMGP97] LOFFELMANN H., MROZ L., GROLLER E., PURGATHOFER W.: Stream Arrows: Enhancing the Use of Streamsurfaces for the Visualization of Dynamical Systems. *The Visual Computer 13* (1997), 359–369. 2

[LS07] LI L., SHEN H.-W.: Image-Based Streamline Generation and Rendering. *IEEE Transactions on Visualization and Computer Graphics 13*, 3 (2007), 630–640. 2

[LWSH04] LARAMEE R., WEISKOPF D., SCHNEIDER J., HAUSER H.: Investigating Swirl and Tumble Flow with a Comparison of Visualization Techniques. In *Proceedings IEEE Visualization 2004* (2004), pp. 51–58. 1, 2

[MCHM10] MARCHESIN S., CHEN C.-K., HO C., MA K.-L.: View-Dependent Streamlines for 3D Vector Fields. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010). 2

[MH02] MÖLLER T., HAINES E.: *Real-Time Rendering*, 2 ed. A. K. Peters Limited, 2002. 7

[MLP*10] MCLOUGHLIN T., LARAMEE R. S., PEIKERT R., POST F. H., CHEN M.: Over Two Decades of Integration-Based, Geometric Flow Visualization. *Computer Graphics Forum 29*, 6 (2010), 1807–1829. 1, 2

[MMWC11] MALKI R., MASTERS I., WILLIAMS A. J., CROFT T. N.: The Influence of Tidal Stream Turbine Spacing on Performance. In *Procedings of the 9th European Wave and Tidal Energy Conference (EWTEC)* (Southampton, England, UK, 2011). 8, 9

[MT*03] MATTAUSCH O., THEUSSL T., , HAUSER H., GRÖLLER E.: Strategies for Interactive Exploration of 3D Flow Using Evenly-Spaced Illuminated Streamlines. In *Proceedings of the 19th Spring Conference on Computer Graphics* (2003), pp. 213–222. 2

[Par82] PARASCHIVOIU I.: Aerodynamic loads and performance of the Darrieus Rotor. *J. Energy 6*, 6 (1982), 406–412. 9

[PGL*12] PENG Z., GRUNDY E., LARAMEE R. S., CHEN G., CROFT N.: Mesh-Driven Vector Field Clustering and Visualization: An Image-Based Approach. *IEEE Transactions on Visualization and Computer Graphics* (2012), Forthcoming. 2, 8

[PL09] PENG Z., LARAMEE R. S.: Higher Dimensional Vector Field Visualization: A Survey. In *Theory and Practice of Computer Graphics (TPCG '09)* (June 2009), pp. 149–163. 1

[PS09] PEIKERT R., SADLO F.: Topologically Relevant Stream Surfaces for Flow Visualization. In *Proc. Spring Conference on Computer Graphics* (April 2009), Hauser H., (Ed.), pp. 43–50. 2

[PVH*03] POST F. H., VROLIJK B., HAUSER H., LARAMEE R. S., DOLEISCH H.: The State of the Art in Flow Visualization: Feature Extraction and Tracking. *Computer Graphics Forum 22*, 4 (Dec. 2003), 775–792. 1

[Rot00] ROTH M.: *Automatic extraction of vortex core lines and other line-type features for scientific visualization*. PhD thesis, Technische Wissenschaften ETH Zurich, Zurich, 2000. 5

[SLCZ09] SPENCER B., LARAMEE R. S., CHEN G., ZHANG E.: Evenly-Spaced Streamlines for Surfaces: An Image-Based Approach. *Computer Graphics Forum 28*, 6 (2009), 1618–1631. 2

[TB96] TURK G., BANKS D.: Image-Guided Streamline Placement. In *ACM SIGGRAPH 96 Conference Proceedings* (Aug. 1996), pp. 453–460. 2

[TvW99] TELEA A., VAN WIJK J.: Simplified Representation of Vector Fields. In *Proceedings IEEE Visualization '99* (1999), pp. 35–42. 3, 4

[TWHS03] THEISEL H., WEINKAUF T., HEGE H.-C., SEIDEL H.-P.: Saddle Connectors–An Approach to Visualizing the Topological Skeleton of Complex 3D Vector Fields. In *Proceedings IEEE Visualization 2003* (2003), pp. 225–232. 2

[VBvP04] VILANOVA A., BERENSCHOT G., VAN PUL C.: DTI visualization with streamsurfaces and evenly-spaced volume seeding. In *Joint Eurographics - IEEE TCVG Symposium on Visualization* (2004), Deussen O., Hansen C., Keim D., Saupe D., (Eds.), Eurographics Association, pp. 173–182. 2

[WHN*03] WEINKAUF T., HEGE H., NOACK B., SCHLEGEL M., DILLMANN A.: Coherent Structures in a Transitional Flow around a Backward-Facing Step. *Physics of Fluids 15*, 9 (September 2003), S3. Winning Entry from the Gallery of Fluid Motion 2003. 2

[WSE05] WEISKOPF D., SCHAFHITZEL T., ERTL T.: Real-Time Advection and Volumetric Illumination for the Visualization of 3D Unsteady Flow. In *Data Visualization, Proceedings of the 7th Joint EUROGRAPHICS–IEEE VGTG Symposium on Visualization (EuroVis 2005)* (May 2005), pp. 13–20. 4, 7

[WT02] WEINKAUF T., THEISEL H.: Curvature Measures of 3D Vector Fields and their Application. In *Journal of WSCG* (2002), V.Skala, (Ed.), vol. 10, pp. 507–514. 2

[WTHS04] WEINKAUF T., THEISEL H., HEGE H. C., SEIDEL H.-P.: Boundary Switch Connectors for Topological Visualization of Complex 3D Vector Fields. In *Proceedings of the Joint Eurographics - IEEE TCVG Symposium on Visualization (VisSym 04)* (2004), pp. 183–192. 2

[XI05] XU R., II D. W.: Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks 16*, 3 (May 2005), 645–678. 2

[XLS10] XU L., LEE T.-Y., SHEN H.-W.: An Information-Theoretic Framework for Flow Visualization. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1216 –1224. 2

[YKP05] YE X., KAO D., PANG A.: Strategy for Seeding 3D Streamlines. In *Proceedings IEEE Visualization 2005* (2005), pp. 471–476. 2

[ZSH96] ZÖCKLER M., STALLING D., HEGE H.: Interactive Visualization of 3D-Vector Fields Using Illuminated Streamlines. In *Proceedings IEEE Visualization '96* (Oct. 1996), pp. 107–113. 2