Vision, Modeling, and Visualization (2013)
Michael Bronstein, Jean Favre, and Kai Hormann (Eds.)

Illustrative Rendering of Particle Systems

Jennifer Chandler[1], Harald Obermaier[1], and Kenneth I. Joy[1] †

[1]UC Davis IDAV

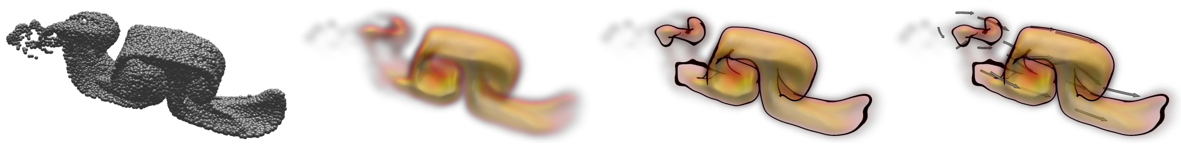

Figure 1: *The far left image shows the underlying point cloud from the von Karman vortex street data set. The next three images show a volume rendering of the point cloud density with no annotations, with silhouette edges, and with silhouette edges and trajectory arrows.*

Abstract

Sets of particles are a frequently used tool for the exploration of time-varying flow fields due to their ease of use and conceptual simplicity. Understanding temporal changes in such particle systems can be difficult with traditional visualization methods such as isosurface rendering and particle splatting. These types of methods only show the current shape of the point cloud and give no context about how the current time step relates to previous or future time steps. In this paper we present an illustrative rendering approach to visualizing particle systems. We use illustrative rendering techniques like silhouettes and trajectory arrows combined with volume raycasting to highlight important features in the particle system and show how these features change across time steps. Our method allows users to easily identify structures within the point cloud and understand how they evolve over time.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.3]: Three-Dimensional Graphics and Realism—Color, Shading, Shadowing, and Texture Simulation and Modeling [I.6.6]: —

1. Introduction

Applying illustrative rendering techniques to visualization applications has been a growing area of interest. Drawing styles used in illustration, as often found in textbooks and instruction manuals, aim to generalize information about the object or process depicted while at the same time drawing attention to specific regions or features of interest. Applying the techniques used in illustration to scientific visualization can help convey important insights about the data while minimizing visual distractions from uninteresting regions of the data. These types of visual abstractions become especially important in data sets where there are a large number of data points, as visualizing all of the data with equal emphasis can generate confusing and uninformative visualizations. Particle systems can generate large amounts of data with high frequency details which can be difficult to interpret using traditional visualization techniques, which make them a good candidate for illustrative visualization approaches.

A particle system can be thought of as a set of point clouds that changes over time. Analyzing these point clouds both individually and collectively can be challenging with common methods of point cloud visualization, especially with dense point clouds. Visualizing only the points can occlude important information from the view about the high frequency details as the points move in the center of the point cloud. Rendering the surface of the point cloud can show exterior features but will miss features developing in the interior. Rendering isosurfaces of point density can show features within the point cloud but does not provide a context to how these features relate to the rest of the point cloud.

† email: {jchandler, hobermaier, kijoy}@ucdavis.edu

DOI: 10.2312/PE.VMV.VMV13.177-185

In our work, we use a volume rendering approach based on a *Kernel Density Estimation* (KDE) of the point cloud to give insight into the interior of the point cloud as well as providing a context for how different regions of the point cloud relate to one another. Displaying silhouette edges based on the density gradient further emphasizes features within the point cloud. The volume rendering of the point cloud only provides information about a single time step of the particle system at a time, but lacks the ability to track changes in the point cloud. Looking at sequential volume renderings can give some insight to how the point cloud evolves, but it can be difficult to track features visually when viewing single images. Adding trajectory arrows to the volume rendering shows how features move over time and provides a temporal context to a single image of the point cloud.

In this paper we present a method for visualizing time-varying point clouds using an illustrative rendering approach. This allows us to highlight important features and track their movement while minimizing visual clutter inherent in other methods of point cloud visualization. These types of time-varying point clouds can be generated, for example, by seeding particles in a flow field and observing their positions at each time step. It can be difficult to observe properties of the flow field such as separation, convergence, and recirculation by looking at only the set of points at each time step. Applying illustrative rendering techniques to the point clouds can help visualize this behavior.

The remainder of this paper is organized as follows. In Section 2 we provide a background on illustrative rendering. In Section 3 we review prior work in the area of illustrative rendering applied to volume and flow visualization. In Section 4 we discuss the implementation of our illustrative rendering method for particle systems. In Section 5 we discuss the results of applying our method. Finally, we summarize our contributions and give suggestions for future work in Section 6.

2. Background and Techniques

Time-varying flow fields are complex dynamical systems, whose evolution and transport properties are studied in a wide range of application areas, such as climate research and airplane engineering. In order to understand and analyze the dynamic behaviors present in the field, visual exploration techniques may be employed. Such exploration techniques include single trajectory visualization, surface-representation, and the extraction of complex volumetric geometry. Point clouds that are released into the flow field and animated as they traverse the domain represent one of the conceptually most simple (interactive) exploration techniques [SBK07]. For an efficient exploration of the flow field, a point cloud has to be sufficiently dense to be able to capture relevant details of the flow field behavior. Such dense dynamic point clouds have several properties that make efficient visualization challenging:

- *High Frequency Details:* Direct visualization of point clouds results in high frequencies in image space due to large numbers of small objects and a lack of general point cloud continuity.
- *Varying Point Densities:* Divergent and convergent behaviors in the point cloud can lead to strongly varying point densities. These variations are important for the analysis of flow fields, but are not immediately visible in classic point-based renderings.
- *Complex Movements and Interaction:* Time-varying point-clouds often show complex relative movements and particle interactions. Such relative motion can result in the point cloud breaking up into multiple distinct clouds, merging, or reorganizing. This seemingly chaotic behavior together with the lack of a parameterization of the point cloud makes the perception of the behavior of the point cloud over time difficult.

In this paper we aim to develop illustrative rendering techniques that overcome these visualization challenges and allow for a more complete analysis of time-varying point clouds or particle systems. Illustrative rendering encompasses a wide range of techniques that can be applied to visualization to emphasize important information about the underlying data and minimize parts that do not enhance the overall understanding of the data. In their overview of illustrative visualization approaches and techniques Rautek et al. [RBGV08] argue that the phrase illustrative visualization is a tautology since all visualization is illustrative, and both illustration and visualization have the same goal of enhancing understanding of the underlying data. They classify illustrative techniques as falling into either a high level or low level category. High level illustrative visualization techniques focus on what parts of the data to view and how to emphasize it, while low level techniques focus on the visual style such as shading techniques, outlining, and texturing. Many applications of illustrative rendering to scientific visualization use a combination of both high level and low level illustrative techniques best suited to the underlying data and the information to be conveyed about the data. In our time-varying point cloud data we want to emphasize features that develop due to particle movement in the context of the entire point cloud as well as showing how these particles move within the point cloud over time. To meet these requirements in our application of illustrative rendering techniques we use high level techniques like trajectory arrows (speed lines) and regions of emphasis in combination with low level techniques such as silhouette edges and multiple shading models.

2.1. Point Cloud Abstraction

In a large particle system, properties of an individual particle are significantly less important than the behavior of the point cloud as a whole. In illustrative visualization such a prioritization and abstraction is a well-known and frequently used concept, in which repetitive, small scale details or objects are sacrificed for the improved perception of the continuum. In the design of our visualization system we abstract from the notion of single particles by transferring the particle system into a density representation as shown in Figure 2. This representation is suitable for the application of volume rendering techniques and at the same time reduces high frequencies present in point-based rendering.

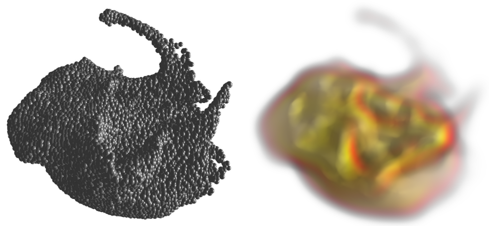
Figure 2: *A view that abstracts from individual points in a set (left) is given by a cloud-like representation (right).*

2.2. Regions of Emphasis

In many technical illustrations certain parts of the object may be more important or of more interest. These regions are often emphasized using techniques such as magnification, change of focus, or cut-away views. In volume rendering, regions of emphasis are especially important because there can be many overlapping objects or features within the volume which can obscure or occlude each other. To emphasize the features of interest in volume rendering, techniques such as cutting planes which act as windows to the interior of the volume or opacity modulation based on depth or feature characteristics can be used to focus the viewer on the region of emphasis while still remaining in the overall context of the volume. These techniques are often referred to as "focus + context" techniques [RBGV08]. We choose to use opacity modulation based on a user defined region of emphasis. Because we use the point cloud density as the opacity we allow the user to select a region of density values to emphasize. Density values that fall outside of this region have their opacities decreased so that the region of emphasis is more apparent.

2.3. Silhouette Edges

Silhouette edges provide an outline for an object based on the viewing direction. These types of edges are used in a wide variety of illustrations and cartoons to show boundaries of objects and distinguish objects from one another. Methods for detecting silhouette edges can be either image-based or object-based [IFH*03]. Image-based methods extract edges based on color variations in the rendered image. This can be a computationally efficient approach because it depends only on the final output image and not on any intermediate geometry or rendering steps. Object-based methods for silhouette edge detection use the angle between the normal vector of the object and the viewing vector to detect the parts of the object that are nearly perpendicular to the view. We use an object-based method for edge detection because it allows us to easily detect edges that occur in the interior of the volume, which could be lost in an image-based edge detection method if the feature is partially obscured by other parts of the volume. Adding silhouette edges to the volume rendering helps to further emphasize features within the volume that may be difficult to detect when they are rendered behind other partially transparent parts of the volume. Figure 4 shows the silhouette edges in a time step of the jet data set.

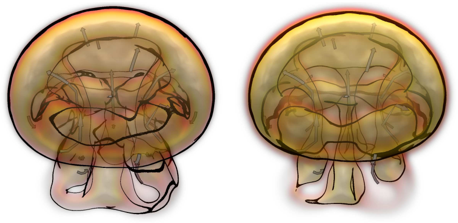
Figure 3: *Shifting the focus or regions of emphasis allows for subtle highlighting of relevant features in the point cloud. The left image emphasizes a low density region of the jet data set, while the right image emphasizes a higher density region.*

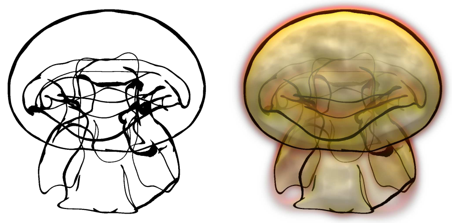
Figure 4: *Silhouette are a central illustration techniques to convey shape properties. In point cloud renderings, this can contribute significantly to the quality of structure perception.*

2.4. Shading Models

The choice of shading model in illustrations can help to enhance perception of important features. Using a physically based shading model such as the Phong model [Pho75] may help users to recognize features within the volume because it attempts to mimic real world shading. However, sometimes information about the shape can be lost in regions of the object facing away from the light source. Often in technical illustrations, a tone-based shading model is used instead such as the cool to warm shading model proposed by Gooch et al. [GGSC98]. This type of shading model blends between cool colors in regions facing away from the light and warm colors in regions facing the light source and is often using in conjunction with silhouette edges to emphasize object shape and boundaries. Both of these lighting models have different benefits and can be used to help perceive the shape of features in volume rendering. Figure 5 compares spheres shaded with these two models. We provide the user with the option to toggle between these two lighting models in real time.

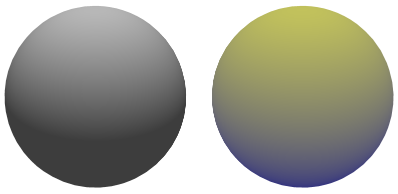

Figure 5: *Shading techniques such as Phong shading (left) and Gooch shading (right) can highlight different properties of the point cloud.*

2.5. Trajectory Arrows

We display trajectory arrows in different regions of the volume rendering to show the movement of particles within the point cloud across time steps. This technique is very similar to the concept of speed lines in cartoons and illustrations [JR05]. Speed lines are thin parallel lines that extend from the object to display the path the object traveled prior to its current position in the image. Due to the nature of the particle system, drawing thin lines to convey movement within the point cloud may be confusing since the lines may overlap depending on the viewing angle which may make it difficult to tell the direction of particle movement. Additionally since we render silhouette edges as thin lines, the two types of lines might be difficult to distinguish from one another. Instead, we keep this concept of showing prior movement within a single image by using trajectory arrows. These arrows correspond to the path of particles in previous time steps. We render flat shaded outlined gray arrows aligned with the view to show the movement of particles within the volume. The advantage of using these arrows is that they are distinct from the silhouette edges and because they have a thickness it is easy to distinguish trajectories that cross paths within the volume due to the viewing angle and the transparent nature of a volume rendering. Figure 6 shows the set of trajectory arrows and how they are integrated with the volume rendering.

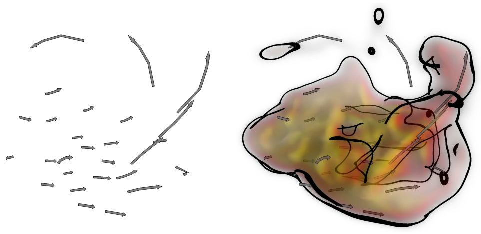

Figure 6: *Trajectory arrows are closely related to speedlines used in illustration and are able to convey a multi-temporal aspect of time varying point clouds.*

3. Related Work

Illustrative rendering has been used in a wide variety of visualization applications to provide a better understanding of the underlying data. We focus our review of related work on illustrative rendering approaches in volume visualization and in flow visualization since these two areas most closely relate to our application of illustrative rendering techniques to particle systems generated from flow field data sets.

3.1. Illustrative Volume Visualization

Illustrative rendering techniques have often been applied to volume rendering because volume rendered images show many objects of varying opacities that can overlap, surround, and occlude each other, necessitating a way to reduce visual clutter and emphasize important features. Bruckner et al. [BGKG05] use a modified version of volume clipping that preserves the context of features as well as modulating the opacity by the light intensity, gradient magnitude, and distance from the view plane to show relevant features in context. Balabanian et al. [BVMG08] use temporal style transfer functions to render all time steps of time-varying volume data in a single image by using different colors and opacities for different time steps and providing multiple viewing modes for regions of overlap such as temporal maximum intensity projection, temporal average, and temporal maximum density change. Csébfalvi et al. [CMH*01] render object contours using gradient magnitude combined with maximum intensity projection and depth cueing. Lum and Ma [LM02] use tone shading and depth cueing along with surface enhancement using gradient magnitude to provide better recognition of objects and their spatial relationships within the volume. Rheingans and Ebert [RE01] use non-photorealistic rendering techniques such as silhouettes, boundary and region enhancement, and depth cueing along with physics-based shading techniques to provide better recognition of features. Bruckner and Gröller [BG07] use style transfer functions based on lit sphere maps to apply different shading techniques to different objects in the volume which can be used to replicate shading styles found in medical illustrations. There are many common aspects of illustrative rendering in these works, some of which we also apply to our own work, like silhouette edges, opacity modulation in regions of interest, and the use of different shading models. Because our volume data is generated based on the density of the points, we tend not to have distinct boundaries around features in the volume, which means we have to modify techniques such as silhouette edge detection to work with smooth transitions between density regions rather than sharp object boundaries.

3.2. Illustrative Flow Visualization

Many different types of illustrative techniques have been applied to flow visualization. For a complete review of illustrative techniques in flow visualization see Brambilla et al.'s state of the art report [BCP*12]. Hsu et al. [HMCM10] use both image space and object space methods to depict the time evolution of flow using composited volume renderings of multiple time steps with silhouette edges and toon shading. Joshi and Rheingans [JR05] use feature extracted volume flow data and track the movement of features over time using speed lines, flow ribbons, strobe silhouettes, and compositing of multiple time steps in a single image. Born et al. [BWF*10] use illustrative techniques such as silhouettes, surface stream lines, moveable cuts and slabs, and halftoning, applying them to stream surfaces. Hummel et al. [HGH*10] apply illustrative techniques to integral surface data by opacity modulation based on surface normals in combination with silhouette edges and texture mapping to indicate flow direction. Svakhine et al. [SJEG05] apply photographic techniques such as Schlieren photography and shadowgraph rendering as well as boundary enhancement, silhouette edges, and illustrative transfer functions for volume rendering. Bürger et al. [BKKW08] render particles in the flow field and modify their appearance and density based on their importance. They also draw cluster arrows to indicate regions of similar flow. Krüger et al. [KKKW05] render particles in the flow field and change the color and shape of the particles to show information about the flow such as direction and integration error. Max et al. [MBC93] render smoke volumes in the flow field by tesselating the volume with tetrahedra. Similar to our approach, rendering a smoke volume can give more insight into flow behavior than rendering just particles or trajectories. Von Funck et al. [vFWTS08] also render smoke volumes in the flow field but use semi-transparent streak surfaces to represent the smoke which is faster than previous approaches because it does not require adaptive mesh restructuring. Bürger et al. [BFTW09] present a method for generating adaptively refined streak surfaces on the GPU and display solid surfaces annoted with stream lines as well as semi-transparent surfaces. Selle et al. [SMC04] develop a method for rendering cartoon style smoke from physically-based smoke particle simulations by rendering particle glyphs and detecting silhouette edges. Crawfis and Max [CM93] use a splatting based approach to volume rendering of flow fields which uses textured splats to incorporate direction and velocity information into the volume rendering. Hauser and Mlejnek [HM03] allow users to define regions of interest and isosurfaces or volume renderings based on these importance values. They also seed streamlines in regions of high importance and fade them out as they enter low importance regions. Like our time-varying point cloud data, flow field visualization also has a need to show changes in the data over time and features that emerge within the data. Thus we can adapt many of these same illustrative techniques to our point cloud visualization such as displaying edges, using arrows or lines to show movement of the data over time, and emphasizing user indicated regions of interest. Many illustrative rendering techniques in flow visualization focus on surfaces or distinct particles in the data, while we apply a kernel density estimation to generate a smooth representation of the entire point set.

4. Illustrative Methods

To demonstrate our use of illustrative rendering techniques we have created an interactive application that allows users to change viewing parameters in real-time. This allows the user to explore different rendering styles and interactively choose regions of emphasis. In a preprocessing step we create a density volume for each time step of the point cloud using *Kernel Density Estimation* (KDE) [WJ95]. This step generates a continuous representation of the point cloud data, which is used as the basis for the volume rendering and silhouette edge detection. In a separate preprocessing step we perform clustering on the trajectories to get a representative set of trajectory arrows. In our volume raycaster [Lev88] we display a volume rendering of a single time step of the point cloud which can be annotated with silhouette edges and trajectory arrows.

4.1. Kernel Density Estimation

We use KDE to create a smooth continuous representation of the underlying point data and minimize high frequency noise due to irregularities in point spacing and small object sizes. We calculate a radius for the KDE by dividing the volume of the initial point cloud (calculated based on the initial particle seeding region) by the number of points and solving for the radius of a single particle. We then multiply this radius by a constant k to increase the kernel width. Larger k will give a smoother overall density volume by blurring the contribution of each point in the cloud, while a smaller radius will emphasize individual points in the cloud. This allows for control over visual abstraction. We have found in practice that a k value of 12 gives a good balance between emphasizing important features of the point cloud while minimizing noise. We assume that the initial seeding of the point cloud is relatively uniform because we use a uniform radius in each direction for the kernel density estimation. We create the density volume by spatially partitioning the points in a k-d tree and performing a kernel density estimation at the center of each voxel in the density volume. We calculate the density ρ at a point p in the volume as the sum of cubic smoothing kernels centered at its neighbors

$$\rho(p) = \sum_{|p-p_i|\leq r} \left(1 - \frac{\|p-p_i\|^2}{r^2}\right)^3 \qquad (1)$$

where $p = (p_x, p_y, p_z)$ is the location of the density estimate, p_i is a particle from the point cloud, and r is the radius of the kernel density estimation. During the density estimation we also calculate a gradient $\nabla\rho(p)$ along each axis for each sample using the derivative of equation 1 as follows:

$$\nabla\rho(p) = \sum_{|p-p_i|\leq r} \begin{bmatrix} \frac{-3}{r^2}\left(1-\frac{\|p-p_i\|^2}{r^2}\right)^2 (2(p_x-p_{ix})) \\ \frac{-3}{r^2}\left(1-\frac{\|p-p_i\|^2}{r^2}\right)^2 (2(p_y-p_{iy})) \\ \frac{-3}{r^2}\left(1-\frac{\|p-p_i\|^2}{r^2}\right)^2 (2(p_z-p_{iz})) \end{bmatrix} \tag{2}$$

The normalized version of $\nabla\rho(p)$ is used in lighting and silhouette edge detection. After calculating the density volumes for each time step we find the maximum density value over all time steps which is used to normalize the density values. The kernel density estimation is performed on the GPU using OpenCL. We construct the k-d tree on the CPU and upload it to GPU memory. We allocate one thread for each density estimate. Each thread traverses the k-d tree to find all particles within the kernel radius of the sample location and calculates the density and gradient which are written to an array of single precision floating point numbers.

4.2. Volume Rendering

To display the point cloud density we use a GPU raycaster written in GLSL. Using a GPU based raycaster allows us to achieve an interactive frame rate so the user can explore each time step in real time. We create a 3D floating point texture with the normalized gradient in the RGB channels and the scaled density in the alpha channel. At each sample along the ray we look up the density value in the 3D texture and calculate a color by using the density value as an index into a color map. We use this color and the normalized gradient with either the Phong [Pho75] shading model or the Gooch shading model [GGSC98] to get a shaded color for the sample. We use the density value ρ scaled by a constant opacity multiplier as the alpha value. We also add an option for a range of emphasis. All density values inside the range of of emphasis (ρ_{min}, ρ_{max}) are scaled by an opacity multiplier α_1 while values outside the range of emphasis are scaled by a linear interpolation of α_1 and an out of range opacity multiplier α_2 which gives the following equations for the opacity value α for a sample.

$$\alpha = \begin{cases} \rho\alpha_1 & \rho_{min} < \rho < \rho_{max} \\ \rho((1-(\rho/\rho_{min}))\alpha_2 + (\rho/\rho_{min})\alpha_1) & \rho \leq \rho_{min} \\ \rho((1-\frac{(\rho-\rho_{max})}{(1-\rho_{max})})\alpha_1 + \frac{(\rho-\rho_{max})}{(1-\rho_{max})}\alpha_2) & \rho \geq \rho_{max} \end{cases} \tag{3}$$

The values α_1 and α_2 that define the range of emphasis are set by the user and can be changed interactively at run time to view different regions of emphasis. We use front to back alpha blending along the ray and stop when the alpha value becomes very close to fully opaque (α = 0.96) or when the ray exits the volume.

4.3. Silhouette Edges

To further emphasize features within the point cloud we render silhouette edges along with the ray casting. If the dot product of the gradient vector and the viewing ray is less than a predefined threshold at a particular sample along the ray then we render the sample as fully opaque black. Because of the kernel density estimation there are no completely solid surfaces within the density volume of the point cloud so rendering all silhouette edges leads to very thick edge lines that obscure most of the volume. Instead we limit display of silhouette edges to the range of emphasis which has the effect of outlining features with similar density, much like displaying an isosurface. Unlike rendering an isosurface, we still render density values outside the range of emphasis, but without silhouette edges.

4.4. Trajectory Arrows

We show the movement of the points within the particle system over time using trajectory arrows. These arrows are based on the particle trajectories in the previous time steps. In order to generate the trajectory arrows there must be a correspondence of particles between time steps. We select a representative subset of the particle trajectories to display in the volume by using a clustering algorithm similar to the approach described by Bürger et al. [BKKW08] to generate their cluster arrows. For the clustering we consider three time steps at a time; however, this method could easily be extended to consider more time steps. We assign a similiarity metric to pathlines based on the sum of the squared distances between the particle positions at each time step in the three time step window. Then we start forming clusters of trajectories by selecting any pathline not in a cluster to serve as the reference and examining all other pathlines that are not already in a cluster. If the similarity score between the reference pathline and the candidate pathline is less than a preset threshold then we add the candidate pathline to the cluster. The threshold is determined separately for each data set since it depends on the distances between particles. The threshold can be changed to allow larger or smaller clusters. Once we have examined all possible candidates we output the reference pathline as the trajectory arrow to be shown in the illustrative rendering for that cluster. Then we choose the next pathline that is not already in a cluster and repeat this process of examining candidate pathlines to build up the cluster. Since we only want trajectory arrows that show movement we filter out any pathlines that are shorter than a preset threshold. We also discard clusters that do not have a preset minimum number of pathlines in them.

To show the arrows inside and intersecting the volume rendering we use a multi-pass rendering approach. In the first pass we render the arrow geometry to a frame buffer. We render the arrows as triangle strips by expanding the arrow width along a vector perpendicular to the direction of the trajectory. We expand the arrow geometry in the vertex shader by first rotating the line corresponding to the particle trajectory by the viewing matrix and then adding the width of the arrow to the rotated line so that the arrow's width is

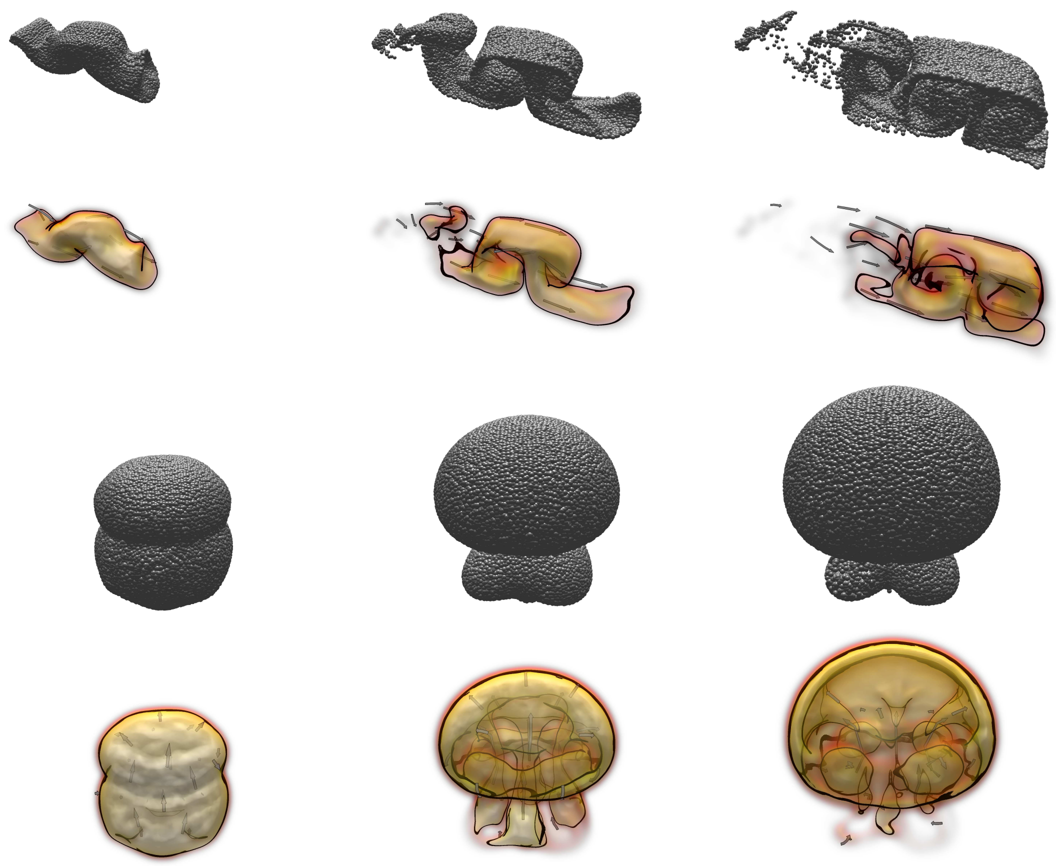

Figure 7: *A comparison image of the point cloud representation and the volume rendering annotated with silhouette edges and trajectory arrows for the von Karman vortex street data set (top) and the jet data set (bottom). By applying illustrative rendering techniques we are able to see how the sparse region of backflow separates from the main flow in the von Karman data set, while in the jet data set the illustrative rendering gives insight into how the particles move in the interior of the volume and create a hollow region in the center.*

always facing the view. After rendering the arrow geometry to a frame buffer, we render a full screen quad containing an image of the arrows to the background of the screen. Then in the raycaster we pass in the depth texture of the arrow geometry. We change the termination condition of the ray to check if the current depth of the sample position along the ray is behind the arrow depth. If so, we stop the ray and blend the current color with the background image of the arrows. Using this method allows the arrows to be blended inside the volume so that it is clear which part of the volume the arrow refers to.

5. Results

We show how our technique works with three different flow data sets and describe the new information we are able to analyze by using these illustrative methods.

5.1. Data Sets

We have applied our illustrative rendering techniques to three different flow fields.

Karman Vortex Street In this typical Karman vortex street data set, a cylindrical obstacle is located in a high Reynolds number flow. Immediately behind the object a region with vortical backflow is formed and further downstream behind the flow obstacle periodic vortex cores appear and move through the flow field. In this data set we seeded $47,785$ particles in a cylindrical region immediately past the flow obstacle.

Jet Stream The bottom of this jet data set contains four high-velocity inflow streams. The four streams propagate fluid mass quickly, pushing a front of high acceleration through the flow field. In the course of this acceleration a

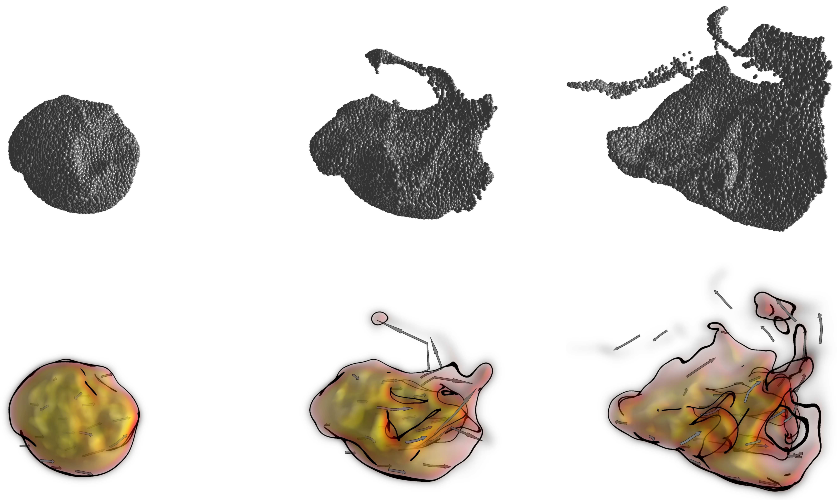

Figure 8: *The dam break data set. The top row shows the point cloud representation. The bottom row shows the volume rendering annotated with silhouette edges and trajectory arrows. Our combination of illustrative rendering techniques shows how the sparse group of particles in the outer shell move up and across the volume while the dense group of interior particles continue moving to the right.*

plume forms. In this data set we injected 296,060 particles in a spherical neighborhood close to the inflow boundary and around the forming plume.

Turbulent Dam Break This is a simulation of a dam break, where a large column of water undergoes strong acceleration and deformation due to gravitational forces. As the water hits the ground, turbulent and chaotic fluid behaviors occur. This behavior is captured by 98,778 particles seeded in a spherical neighborhood close to the surface of the water front.

5.2. Results and Discussion

The top rows of Figure 7 shows three time steps from the von Karman vortex street data set. For these images we used density values between 0.31 and 0.37 with α_1 set to 0.16 and α_2 set to 0.01. The majority of the particles move along the x axis, while there is a small region of back flow near the obstacle in the data set. When visualizing only the points it is difficult to see the higher density structures that form within the volume. Adding the trajectory arrows shows how the forward moving particles separate from the backflow region.

The bottom rows of Figure 7 shows three time steps of the jet data set. For this set of images we chose density values between 0.55 and 0.59 for the range of emphasis with an α_1 value of 0.1 and an α_2 value of 0.01. Looking only at the point cloud representation of this data set does not show any of the structures forming in the interior of the volume. With the density volume representation, it becomes clear that there is a high density ring forming near the center that eventually separates into four smaller dense structures. The silhouette edges help to highlight these interior structures as well as the four separate regions near the bottom of the volume which look like a single structure in the point cloud representation. The trajectory arrows show that there is a strong upward flow through the center of the volume as well as weaker circular flows around the sides.

Figure 8 shows three time steps of the dam break data set. In this set of images we set the range of emphasis to density values between 0.24 and 0.27 with an α_1 value of 0.1 and an $alpha_2$ value of 0.03. The volume rendering of this data set shows that the region in the center of the volume is very dense, while the particles on the outside are comparatively sparse. In particular, in the last time step shown the region near the top, which is moving in the opposite direction of the main flow, is much sparser than it appears to be in the point cloud rendering. The trajectory arrows in this data set show that the particles in the region near the top mainly come from a high velocity region in the outer shell of the volume while the dense region in the center flows generally towards the right.

6. Conclusion

In this paper we presented a method for applying illustrative rendering techniques to particle systems. Using illustrative rendering techniques such as silhouettes, trajectory arrows, and regions of emphasis allows us to reduce visual clutter, highlight important features, and understand how points within the particle system move over time. We demonstrated this method with three different flow field data sets and compared our results to point rendering techniques to show that our method is able to present more relevant information while reducing visual distractions. In the future we would like to explore ways to automatically detect regions of interest instead of relying on user specified parameters. We would also like to apply our illustrative rendering approach to particle systems from other application areas besides flow visualization.

7. Acknowledgments

This work was supported in part by the NSF (IIS 0916289 and IIS 1018097), and the Office of Advanced Scientific Computing Research, Office of Science, through DOE 'Scientific Discovery through Advanced Computing' (SciDAC) contract DE-FC02- 06ER25780, and contract DE-FC02-12ER26072 (SDAV). This work was supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1148897.

References

[BCP*12] Brambilla A., Carnecky R., Peikert R., Viola I., Hauser H.: Illustrative flow visualization: State of the art, trends and challenges. *Proceedings of Eurographics State of the Arts Reports* (2012). 5

[BFTW09] Bürger K., Ferstl F., Theisel H., Westermann R.: Interactive streak surface visualization on the gpu. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1259–1266. 5

[BG07] Bruckner S., GrÃűller M. E.: Style transfer functions for illustrative volume rendering. *Computer Graphics Forum 26*, 3 (2007), 715–724. 4

[BGKG05] Bruckner S., Grimm S., Kanitsar A., Gröller M. E.: Illustrative context-preserving volume rendering. In *Proceedings of EUROVIS* (2005), pp. 69–76. 4

[BKKW08] Bürger K., Kondratieva P., Krüger J., Westermann R.: Importance-driven particle techniques for flow visualization. In *Proceedings of IEEE Pacific Visualization Symposium* (2008), pp. 71–78. 5, 6

[BVMG08] Balabanian J.-P., Viola I., Möller T., Gröller M. E.: Temporal styles for time-varying volume data. In *Proceedings of 3DPVT* (June 2008), pp. 81–89. 4

[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. 5

[CM93] Crawfis R., Max N.: Texture splats for 3d scalar and vector field visualization. In *Proceedings of the 4th conference on Visualization '93* (1993), pp. 261–266. 5

[CMH*01] Csébfalvi B., Mroz L., Hauser H., König A., Gröller E.: Fast visualization of object contours by non-photorealistic volume rendering. *Computer Graphics Forum 20*, 3 (2001), 452–460. 4

[GGSC98] Gooch A., Gooch B., Shirley P., Cohen E.: A non-photorealistic lighting model for automatic technical illustration. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), ACM, pp. 447–452. 3, 6

[HGH*10] Hummel M., Garth C., Hamann B., Hagen H., Joy K. I.: Iris: Illustrative rendering for integral surfaces. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1319–1328. 5

[HM03] Hauser H., Mlejnek M.: Interactive volume visualization of complex flow semantics. In *Proc. of the 8th Fall Workshop on Vision, Modeling and Visualization (VMV 2003)* (2003), pp. 191–198. 5

[HMCM10] Hsu W.-H., Mei J., Correa C. D., Ma K.-L.: Depicting time evolving flow with illustrative visualization techniques. In *Arts and Technology*. Springer, 2010, pp. 136–147. 5

[IFH*03] Isenberg T., Freudenberg B., Halper N., Schlechtweg S., Strothotte T.: A developer's guide to silhouette algorithms for polygonal models. *IEEE Computer Graphics and Applications 23*, 4 (2003), 28–37. 3

[JR05] Joshi A., Rheingans P.: Illustration-inspired techniques for visualizing time-varying data. In *Proceedings of IEEE Visualization* (2005), pp. 679–686. 4, 5

[KKKW05] Krüger J., Kipfer P., Kondratieva P., Westermann R.: A particle system for interactive visualization of 3D flows. *IEEE Transactions on Visualization and Computer Graphics 11*, 6 (11 2005), 744–756. 5

[Lev88] Levoy M.: Display of surfaces from volume data. *IEEE Computer Graphics and Applications 8*, 3 (1988), 29–37. 5

[LM02] Lum E. B., Ma K.-L.: Hardware-accelerated parallel non-photorealistic volume rendering. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering* (2002), pp. 67–74. 4

[MBC93] Max N., Becker B., Crawfis R.: Flow volumes for interactive vector field visualization. In *Proceedings of the 4th conference on Visualization '93* (Washington, DC, USA, 1993), VIS '93, IEEE Computer Society, pp. 19–24. 5

[Pho75] Phong B. T.: Illumination for computer generated pictures. *Commun. ACM 18*, 6 (June 1975), 311–317. 3, 6

[RBGV08] Rautek P., Bruckner S., Gröller E., Viola I.: Illustrative visualization: new technology or useless tautology? *ACM SIGGRAPH Computer Graphics 42*, 3 (2008), 4. 2, 3

[RE01] Rheingans P., Ebert D.: Volume illustration: Non-photorealistic rendering of volume models. *IEEE Transactions on Visualization and Computer Graphics 7*, 3 (2001), 253–264. 4

[SBK07] Schirski M., Bischof C., Kuhlen T.: Interactive exploration of large data in hybrid visualization environments. In *Proceedings of the 13th Eurographics Conference on Virtual Environments* (2007), EGVE'07, pp. 69–76. 2

[SJEG05] Svakhine N. A., Jang Y., Ebert D., Gaither K.: Illustration and photography inspired visualization of flows and volumes. In *Proceedings of IEEE Visualization* (2005), pp. 687–694. 5

[SMC04] Selle A., Mohr A., Chenney S.: Cartoon rendering of smoke animations. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering* (New York, NY, USA, 2004), NPAR '04, ACM, pp. 57–60. 5

[vFWTS08] von Funck W., Weinkauf T., Theisel H., Seidel H. P.: Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1396–1403. 5

[WJ95] Wand M. P., Jones M. C.: *Kernel smoothing*, vol. 60. Chapman & Hall/CRC, 1995. 5