View Point Evaluation and Streamline Filtering for Flow Visualization

Teng-Yok Lee Oleg Mishchenko Han-Wei Shen Roger Crawfis

Department of Computer Science and Engineering
The Ohio State University*

ABSTRACT

Visualization of flow fields with geometric primitives is often challenging due to occlusion that is inevitably introduced by 3D streamlines. In this paper, we present a novel view-dependent algorithm that can minimize occlusion and reveal important flow features for three dimensional flow fields. To analyze regions of higher importance, we utilize Shannon's entropy as a measure of vector complexity. An entropy field in the form of a three dimensional volume is extracted from the input vector field. To utilize this view-independent complexity measure for view-dependent calculations, we introduce the notion of a maximal entropy projection (MEP) framebuffer, which stores maximal entropy values as well as the corresponding depth values for a given viewpoint. With this information, we develop a view-dependent streamline selection algorithm that can evaluate and choose streamlines that will cause minimum occlusion to regions of higher importance. Based on a similar concept, we also propose a viewpoint selection algorithm that works hand-in-hand with our streamline selection algorithm to maximize the visibility of high complexity regions in the flow field.

Index Terms: I.3.3 [Computing Methodologies]: Computer Graphics—Picture/Image Generation

1 INTRODUCTION

When visualizing 3D datasets, having a clear visibility of important features is crucial but occlusion often gets in the way. For 3D flow data, displaying a large number of long and winding particle traces can easily block the view to more important regions, and hence hinder understanding of the data.

To reduce occlusion and highlight salient flow features, existing work can be roughly categorized as view-independent and view-dependent. For view-independent methods [3, 22, 23], the goal is to highlight salient flow features by placing denser streamlines around more important regions. However, since the features are detected in object space, they may still be occluded when viewed from certain viewpoints. To remedy this problem, view-dependent methods [11, 12] control the density of streamlines placed on the screen to minimize occlusion. These methods, however, do not always consider the locations of features due to the lack of a feature identification stage.

Occlusion of flow features can be caused not only by an excessive amount of streamlines, but also by an improper choice of viewpoint. If from a given viewpoint the features are mostly self-occluded, there is very little that the streamline placement algorithm can do to alleviate the occlusion problem. While a number of algorithms have been proposed to find good viewpoints for triangle meshes [19], isosurfaces [17] and volumes [1, 7, 21], very little work has been done to address issues related to the visualization of three-dimensional flow fields.

To effectively visualize salient features in a static flow field, in this paper we propose an algorithm that considers both viewpoint selection and view-dependent streamline placement. To identify salient flow regions, the proposed algorithm computes the entropy of the flow data based on the information-theoretic framework described in Xu *et al.* [22]. The idea behind the framework is that, vectors near regions that contain salient flow features typically distribute more randomly and hence contain higher entropy. Consequently, the complexity of each region in a flow field can be computed as a scalar field called the *entropy field*. From a given image plane, to identify regions with maximal entropy we use the conventional maximal intensity projection (MIP) method to render the entropy field into an image called Maximal Entropy Projection (MEP). We also store the depth value associated with the maximum entropy value for each pixel. We refer to the buffer that stores these two entities as the *MEP-framebuffer*. Once regions relative to the given viewpoint that have higher entropy are located, streamlines are placed accordingly to minimize the occlusion to these regions. We also utilize the view-dependent projection of entropy values to search for an optimal viewpoint.

This paper has two distinct contributions. First, we introduce a novel streamline selection algorithm that ensures good visibility for salient flow features from a given viewpoint. Second, we present an algorithm that can find the optimal viewpoint for three-dimensional vector fields. To the best of our knowledge, our paper is the first that considers viewpoint selection and streamline placement simultaneously.

This paper is organized as follows. After related work is reviewed in Section 2, the relationship between flow features and flow complexity and the MEP framebuffer is presented in Section 3. Sections 4 and 5 present our view-dependent streamline placement and viewpoint selection algorithms. Comparison between our algorithms and related approaches is presented in Section 6, and the implementation and performance of the tests is provided in Section 7. This paper is concluded with future work in Section 8.

2 RELATED WORK

The number of papers on flow visualization is very high as it has been and continues to be an area of active research. For a good overview we refer the reader to state of the art reports by Hauser *et al.*, [6], Laramee *et al.* [10], and McLoughlin *et al.* [13]. Among this research, many approaches have been developed for streamline placement in two dimensional [8, 14, 18, 20] and three dimensional vector fields [3, 5, 11, 12, 22, 23]. Since our goal here is to visualize 3D vector fields, we will focus on those techniques for 3D domain.

Chen *et al.* [3] propose a method that employs a similarity metric for placing streamlines, which can be used for both two- and three-dimensional visualization. Ye *et al.* [23] extract critical points from a vector field and generate view-independent streamlines by applying specific seeding patterns that depend on types of critical points. Marchesin *et al.* [12] rank streamlines for a particular view according to their curvatures and occlusion that an individual streamline adds. To evaluate streamline curvature, they introduce a notion of streamline angular entropy similar to streamline entropy of [5]. Li and Shen [11] discuss image-based streamline placement. Instead of placing streamlines in three-dimensional space they first analyze projections of streamlines to a user-specified 2D surface and then unproject those streamlines that do not cause clutter and occlusion

*e-mail:leeten,mishchen,hwshen,crawfis@cse.ohio-state.edu

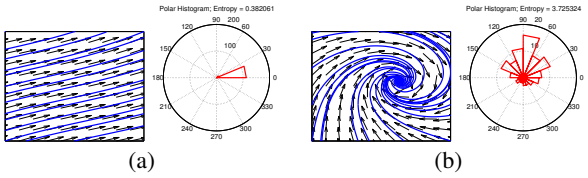


Figure 1: Entropy and complexity in flow fields. (a): A vector field with homogeneous flow and the polar histogram of the vector orientations. (b): A more complex flow field with its polar histogram. The entropies are measured from histograms of 16 bins, and thus the range is $[0, log_2(16)]$. The entropies for (a) and (b) are 0.38 and 3.73, respectively.

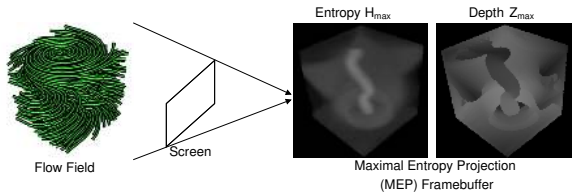


Figure 2: Illustration of the MEP-framebuffer. A MEP-framebuffer consists of two buffers H_{max} and Z_{max} that, for a given viewpoint, store the maximal entropies for all pixels and the corresponding depths of the voxels with maximal entropy.

in 3D. Recently Spencer *et al.* proposed an image-based method to place streamlines evenly on 3D surfaces [16].

In recent years, information theory has been applied to various problems in computer graphics and visualization, where different information-theoretic approaches are proposed [2, 22]. Our work is based on the framework in [22], which extracts regions of high complexity from a vector field and places more streamlines there in a view independent manner, while the algorithms in this paper are view-dependent.

Viewpoint selection algorithms for volume rendering and geometric meshes have been studied extensively. Bordoloi and Shen [1] utilize information theory to evaluate views for volume rendering. Takahashi *et al.* [17] decompose volumetric data into feature subvolumes and then use the views that are optimal for these subvolumes to find the best global view, where the isosurfaces are used as the feature descriptor. Viola *et al.* [21] search for optimal viewpoints for volume data based on mutual information between a view and volume dataset features. Kohlmann *et al.* [9] demonstrated a system that synchronizes selecting features in 2D slices with the best corresponding view of a three-dimensional medical volumetric dataset. For polygon meshes, Vázquez *et al.* [19] measure the distribution of the projected mesh polygons to evaluate the complexity of the views. For three-dimensional vector fields, no study has addressed the question about finding the optimal viewpoint yet.

3 Flow Field Complexity

The key component of the algorithms in this paper is the notion of flow data complexity. It is directly related to identifying the locations of regions that have more complex flow patterns. Recently, Xu *et al.* [22] proposed a framework using information theory for quantitative measurements of data complexity in vector fields. Their work also includes a view-independent streamline placement algorithm as the result of information analysis. In this paper we apply a similar measure but focus on view-dependent streamline placement and viewpoint selection. In the following subsections, we first give a general overview of information theory for vector data, and then present the concept of using MEP framebuffers for view-dependent computation.

3.1 Information theory

In information theory, complexity of a distribution is measured by its *entropy*. Given the probability distribution function $P(X = x_i)$, $i = 1 \ldots n$, for a discrete random variable X, Shannon's entropy, denoted as $H(X)$, for this distribution, can be computed as in Equation 1:

$$H(X) = -\sum_{i=1\ldots n} p(x_i) log_2 p(x_i) \quad (1)$$

Shannon's entropy has two important properties. First, the range of the entropy is $[0, log_2(n)]$. As a result, by dividing $H(X)$ by $log_2(n)$, we can get a normalized entropy in the range of $[0, 1]$. Second, and more important, Shannon's entropy is maximal when all outcomes have equal probabilities and is minimal when only one outcome has a non-zero probability.

We can apply the concept of entropy to measure the complexity of vector data: if in a vector field the vectors are all pointing to a similar direction, as shown in Figure 1 (a), the complexity of this field is considered low. On the other hand, if the vector directions spread across many different directions, as shown in Figure 1 (b), the flow field is more complex. Based on this concept, we can measure the complexity of vector data in a local region: for each point in a vector field, the distribution in its neighborhood can be approximated by a histogram of vector orientations in the area. Entropy of this distribution is assigned as the flow complexity at this point. When we compute the entropy score for every data point, we obtain a scalar field called *entropy field*. In [22], Xu *et al.* have empirically demonstrated that entropy in the regions near certain flow features, including critical points and separation lines, is higher than that of other regions. This implies that the entropy field can indicate saliency in the corresponding flow field. Consequently, to effectively visualize salient regions in a flow field, streamlines should be placed and viewpoints should be selected in such a way that the visibility of regions with higher entropy is maximized.

3.2 Maximal Entropy Projection Framebuffer

The entropy field described above represents flow complexity in object space. However, to evaluate viewpoints and minimize occlusion caused by an excessive amount of streamlines, a view-dependent measure of the flow complexity is needed. Such a measure, however, could be nontrivial to calculate, since each pixel in the screen can have contributions from multiple voxels in the entropy volume. Since our goal is to visualize complex flow inside a vector field, we define view-dependent flow complexity as the maximal flow complexity visible from a given viewpoint. More specifically, when visualizing a flow field, a user should be able to see the most complex regions of the flow after the 3D to 2D projection takes place.

Based on this idea, we measure view-dependent flow complexity using a conventional maximal intensity projection (MIP) developed for volume rendering. For each pixel (x, y) on the screen, a ray is cast into the entropy volume and the voxel with the maximal entropy is searched along the ray. The entropy value of the voxel is stored in the corresponding pixel $H_{max}(x, y)$ in the Maximal Entropy Projection (MEP) entropy buffer, and its corresponding depth $Z_{max}(x, y)$ is stored in the MEP z-buffer. We refer to these two buffers as the *MEP-framebuffer*. Figure 2 illustrates the MEP-framebuffer, which displays the two buffers as grayscale images.

We utilize the MEP-framebuffer for view-dependent streamline placement and viewpoint selection, described in the following sections.

(a) (b) (c)

Figure 3: View-dependent streamline evaluation for *Tornado*. (a): The initial set of streamlines where the blue and red ones have negative and positive scores, respectively. (b): The streamlines with positive scores. (c): The streamlines with negative scores.

4 View-dependent Streamline Placement

Before we select suitable streamlines for any given view, a pool of candidate streamlines is first computed from a set of seeds. The criterion to generate such a pool is to have enough streamlines that will sufficiently cover the domain, hence different streamlines from the pool will be used for different view points. In this work, we use the seeding algorithm proposed by Xu *et al.* [22] to generate a pool of streamlines since the algorithm there provides a stopping criteria based on the information content of the selected streamlines. We note that it is possible to use other seeding methods as long as the resulting streamlines can properly cover the entire vector field. After the streamline pool has been created, for a given view our view-dependent streamline placement algorithm is invoked. The placement algorithm consists of two steps. In the first step, each streamline in the pool is evaluated based on whether it occludes the regions that have higher entropy scores. Then, we scan through the streamline pool to select streamlines that cause minimum occlusion. In the following subsections, we explain our algorithm in detail.

4.1 Streamline Evaluation

To evaluate the streamlines for a given view, a scalar value ω_s for each streamline s is computed to decide the priority of a streamline. A streamline has a higher priority if it reveals more complex flow features and a lower priority if it causes occlusion.

To compute ω_s, a score ω_f for each fragment f in the image plane along the streamline s is calculated. The sum of all fragments for the streamline defines the overall score as:

$$\omega_s = \sum_{f \in s} \omega_f \tag{2}$$

For a fragment with object coordinates (x_o, y_o, z_o) and window coordinates (x_w, y_w, z_w), the entropy of the region in object space occupied by the fragment is denoted as $H(x_o, y_o, z_o)$. The score ω_f is computed in such a way that its value is higher when the depth of this fragment z_w is closer to the depth in the MEP framebuffer $Z_{max}(x_w, y_w)$. This is because in this case the fragment is closer to the most complex region along the view ray. The value of ω_f, on the other hand, should be lower if z_w is smaller than $Z_{max}(x_w, y_w)$, since this fragment can cause occlusion to the more complex region.

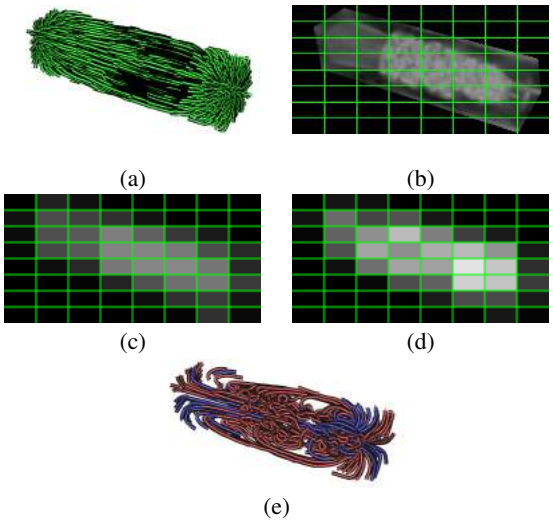
More specifically, if z_w is smaller than $Z_{max}(x_w, y_w)$, the value of ω_f is computed as ω_{front} in Equation 5.

$$\triangle H = H_{max}(x_w, y_w) - H(x_o, y_o, z_o) \tag{3}$$

$$\triangle Z = z_w - Z_{max}(x_w, y_w) \tag{4}$$

$$\omega_f = \omega_{front} = -H_{max}(x_w, y_w)(1 - e^{-|\triangle H||\triangle Z|}) \tag{5}$$

where $H_{max}(x_w, y_w)$ is the maximum entropy along the current view ray, and $\triangle H$ and $\triangle Z$, respectively, are the difference of entropy and depth from the this fragment to the values in the corresponding pixel in the MEP framebuffer. Since this fragment can cause occlusion, its weight is negative so it can reduce the score of this streamline, while the amount of reduction in the score depends on the difference of depth and the entropy. If the depth or entropy of this fragment are near to those in the MEP framebuffer, it is near the salient flow feature that is visible from this pixel and thus the reduction should be small. We design this equation such that the reduction is zero when $\triangle H$ or $\triangle Z$ is zero, close to $H_{max}(x_w, y_w)$ when $|\triangle H|$ and $|\triangle Z|$ is very large.



(a) (b) (c) (d) (e)

Figure 4: Placing streamlines using entropy density. (a): Original set of streamlines. (b): The MEP framebuffer. (c): The expected streamline density for each screen tile, which is equal to the average entropy in the corresponding tile in the MEP framebuffer. (d): Streamline density after streamline placement. (e): The placed streamlines. Red streamlines have nonnegative scores, while blue have negative scores.

For the fragments that are behind the maximum entropy point, i.e., the depth of the fragment z_w is larger than or equal to $Z_{max}(x_w, y_w)$, since they do not cause occlusion, the score ω_{back} is computed as in Equation 6.

$$\omega_f = \omega_{back} = H(x_o, y_o, z_o) e^{-|\triangle H||\triangle Z|} \tag{6}$$

where $\triangle H$ are $\triangle Z$ are defined in Equations 3 and 4, respectively. While ω_{back} is always positive, it becomes smaller as the fragment moves farther away from the point of maximal entropy, or as the difference between the entropy values becomes larger. Meanwhile, similar to ω_{front}, ω_{back} considers the entropy of the fragment and receives a lower score if the entropy is low.

The proposed streamline score has two properties. First, this score can be either positive or negative. Second, and most importantly, the score indicates whether this streamline helps reveal the flow near the salient flow features or cause occlusion. For streamlines with higher scores, it means that these streamlines can either show more information near the flow feature or cause less occlusion than those with lower scores. For instance, Figure 3 shows the streamlines evaluation for the dataset *Tornado*, in which it can be seen that the streamlines with positive scores are around the eye of the tornado or at the bottom of the flow since they do not cause occlusion to the tornado eye, while the streamlines with negative scores can cause occlusion to the tornado eye.

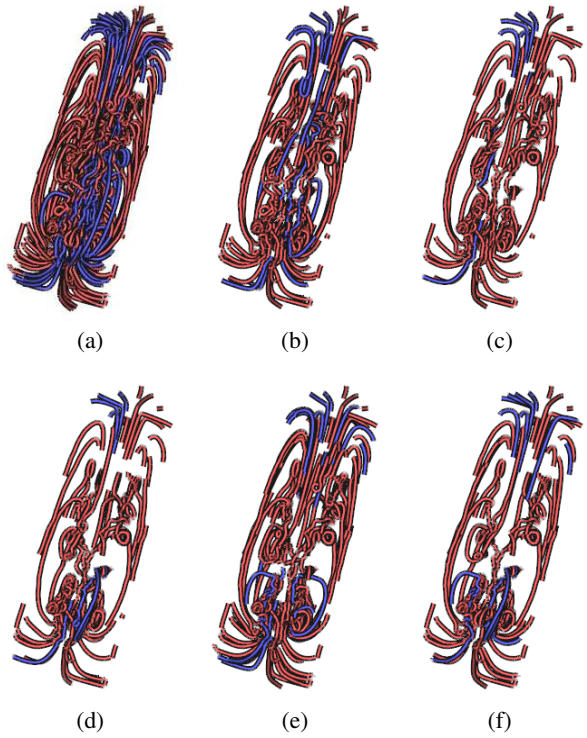


Figure 5: Streamline placement using different line widths. Line width changes from 1 in (a) to 4 in (d). Number of generated streamlines is 181, 85, 59 and 52, respectively. Meanwhile, streamline selection with even density. (e): 87 streamlines with width $w = 6$. (f) 59 streamlines with width $w = 10$. The resolution of all images is 192×384 pixels and 16×16 tiles were used.

4.2 Entropy-dependent Streamline Selection

The scores we compute for the streamlines provide a ranked order over the entire set of possible streamlines. For this, one simple way to select the streamlines is to discard streamlines that have scores lower than a threshold, as shown in Figure 3 where the threshold was set to zero. However, specifying a threshold might not be trivial and the value of the threshold can influence the results dramatically, which can be seen in the accompanying videos. Here we present a fully automatic entropy-based streamline selection algorithm. The algorithm not only minimizes occlusion, but also reveals areas of high complexity in the field.

We design our streamline selection algorithm such that the density of streamlines in a given region on the screen is proportional to the complexity of the flow projected to this area. In other words, areas with more complex flow patterns in image space should display more streamlines. To realize this idea, for a given screen area, we can estimate the *streamline density* by dividing the number of pixels occupied by the streamlines by the total area of the region in pixels. We can also compute for each region on the screen an *expected streamline density*, which is defined as the average normalized entropy of the region in the MEP framebuffer. A region is called *overflowed* if the streamline density exceeds the expected streamline density, or *underflowed* otherwise.

To decide whether a new streamline should be selected, the local neighborhood around the pixels along the streamline is checked. If the underflowed region in the neighborhood in terms of the number of pixels is larger than the overflowed region, the streamline is selected, otherwise it is skipped.

To implement the idea discussed above, we use a simplified algorithm that involves less cost, as illustrated in Figure 4. First of all, screen space is divided into tiles, as shown in Figure 4 (b), and the expected streamline density for each tile is computed as the average entropy of the MEP framebuffer, shown in Figure 4 (c). Given a streamline, screen tiles are checked to see whether the streamline density is going to change if the streamline is placed. Among these tiles, if there exist more underflowed tiles than overflowed tiles, the streamline is allowed to be placed, and streamline densities for all occupied tiles are updated. Figure 4 (e) shows the final result of our algorithm for the streamlines used in Figure 4 (a) with 8×8 tiles, and Figure 4 (d) shows corresponding streamline density for all tiles. It can be seen that in Figure 4 (e) few of the selected streamlines have negative scores. The reason is that as all nonnegative streamlines have been examined but the required density has not been achieved, streamlines with negative scores are allowed to be placed until the density is reached.

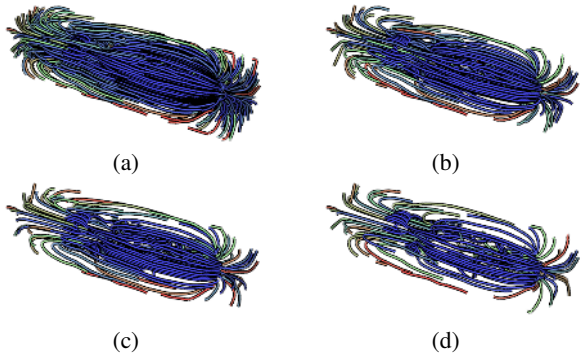


Figure 6: Streamline placement using reverted order. Streamlines with lowest scores are placed first and thus important regions are occluded. Line widths are 1 in (a) to 4 in (d).

It is worth mentioning that this streamline selection algorithm may generate different results if the streamlines are processed in different order. Streamlines that are checked earlier will have a higher probability to be selected. Therefore, to place streamlines that can highlight more complex regions with less occlusion, streamlines with higher scores should be checked first. Otherwise, streamlines that are selected earlier can preclude streamlines even with higher importance from being selected. Examples of such cases are shown in the following subsection.

4.3 Discussion

When using the above streamline selection algorithm, several parameters can alter results. This subsection discusses the effect of the parameters with various examples. These examples use the dataset *Plume* from the same viewpoint. The images have 384×192 pixels, and are divided into 32×32 tiles unless explicitly mentioned. Streamlines are colored according to the sign of their score. The negative, zero, and positive scores are respectively mapped to blue, green, and red.

Streamline width When updating the streamline density for the image tiles, the amount of density contributed by the new streamline can be modulated. In our algorithm the user can specify the width of the streamline to be drawn. When width w is larger than one, a streamline can contribute w times more pixels to the occupied region. To prevent more streamlines from being placed, the expected streamline density for all regions should be divided by w if so desired. Figures 5 (a) - (d) present the streamlines with different line widths. This example shows how streamline density is changed on the screen.

Even versus uneven streamline density By changing the expected streamline density of each tile to a constant value in-

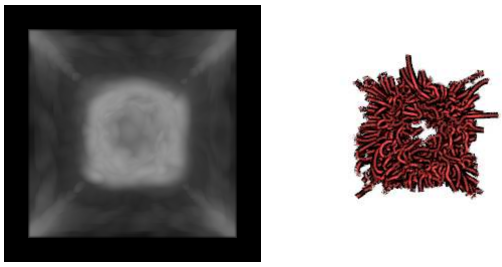


Figure 7: Streamline evaluation from a poor viewpoint for *Plume*. (a): The MEP framebuffer. (b): Streamlines with positive scores. Since regions of high-complexity are self-occluded, the flow in the complex regions cannot be effectively represented even after removing low-scoring streamlines.

stead of the average entropy in the MEP framebuffer, our streamline placement selection can distribute streamlines more evenly on the screen without any changes. In Figures 5 (e) and (f), we show streamlines selected by our algorithm, with even streamline density and width w equal to 6 and 10, respectively. By comparing Figure 5 (e) against Figure 5 (b), we can see that fewer streamlines are placed near the turbulent eddies when using constant streamline density in Figure 5 (e) even though the numbers of streamlines in both figures are close. Similarly, Figure 5 (f) has fewer streamlines near the eddies than Figure 5 (c) in spite that both figures have the same number of streamlines.

Streamline order As mentioned in the Section 4.2, the order in which streamlines are placed can significantly influence the result. To demonstrate this, in Figure 6 we show an example when the algorithm processes streamlines with the lowest scores first. From the color of the placed streamline, it can be seen that most of the selected streamlines have negative scores, causing occlusion to the region of more complex flow and thus generating poorer visualization of the flow field.

5 Finding Optimal Viewpoints

While the view-dependent streamline selection algorithm presented in the previous section can increase the visibility of the more important regions, they can still be unseen if a poor viewpoint is used. Figure 7, for instance, shows the streamlines with top 20% scores for *Plume* from a poor viewpoint, which reveals very little about the actual flow. To remedy the problem, selecting an optimal viewpoint to visualize the streamlines is crucial.

The meaning of *optimality* when referring to viewpoints for a vector field dataset requires some clarification. When the user tries to find the best viewpoint, intuitively s/he would like to see as many interesting features of the flow, preferably with minimum clutter and occlusion. The algorithm that we propose is based on the observation that the larger the area of complex regions projected to the screen, the better those features are visible to the viewer.

From this idea, the optimality of a viewpoint can be evaluated based on the complexity of the projected flow on the screen. This can be analyzed using the previously described MEP-framebuffer. In the MEP-framebuffer, each pixel records the maximal entropy encountered by a viewing ray. The sum of the entropy values in the MEP-framebuffer can provide an upper bound of the complexity, and the mean entropy can provide an average complexity visible from each pixel. As a result, we can quantantively measure the quality for a viewpoint based on the statistics computed from the MEP-framebuffer. Since our goal is to maximize the flow complexity visible to the viewer after projection, we choose the sum of the entropy values as the score.

To search for the optimal viewpoint, we first form the domain of the viewpoints, which is assumed to be a sphere centered at the center of the flow field. This sphere is then tessellated into uniform triangles, where each viewpoint is placed at the center of a triangle. In our experiments, 780 viewpoints across the viewing sphere were used. Once the entropy sum in the MEP-framebuffer for each viewpoint has been computed, the viewpoint with the highest score is then selected as the optimal viewpoint.

It should be noted that there can exist multiple good viewpoints for a given flow field. The leftmost column in Figure 8, for instance, display the scores of the viewpoints for the dataset *Tornado*, where the color from blue to red represents the score from low to high. We can see that more than one viewpoint receive high scores. In the three examples we present, the center of the sphere in each of the images is used as the viewpoint to generate the corresponding images in the following columns. From the image, it can be seen that the scores of the viewpoints are smoothly changed, which is consistent with our expectation in terms of spatial coherence.

Visualizations from better and worse viewpoints for the dataset *Tornado* [4] are compared in Figure 8. As mentioned earlier, better viewpoints reveal more information about the flow, which implies that the image should display better depth cues to highlight the shapes of the streamlines. From the images in the top two rows of Figure 8, it can be seen that there exist two types of bad viewpoints for this dataset. One type of viewpoints display the data from the side, as shown in Figure 8 (a). In Figure 8 (a-1), a blue stripe surrounds the tornado, where the corresponding viewpoints do not allow the viewer to see the streamlines that have high curvature. The other type of bad viewpoint is the one that views the streamlines from the top, as shown in Figure 8 (b-2). Although from this viewpoint one can visualize the streamlines with high curvature, the depth cue is lost. By comparing with Figure 8 (a-3) and (c-3), the MEP-framebuffer in (b-3) shows that the region of high entropy is self-occluded. The bottom row of the figure represents a good viewpoint for this dataset. Compared to the bad viewpoints, Figure 8 (c-2) shows the viewpoint with the highest score, which can display not only the streamlines with high curvatures but also display the 3D nature of the dataset quite well.

The rightmost column in Figure 8 shows the streamlines selected by our streamline selection algorithm for the corresponding viewpoints. While visual clutter and occlusion are reduced compared to using all streamlines as shown in the second column, because of the use of non-optimal viewpoints the results in Figure 8 (a-4) and (b-4) are not as good as in Figure 8 (c-4).

6 Comparison

This section compares our algorithms with related techniques. The streamline evaluation measurement described in Section 4.1 is compared with the metric used by Marchesin *et al.* in their view-dependent streamline placement algorithm [12]. Given a set of streamlines, their metric assigns scores to all streamlines based on their shape complexity and the degree of cluttering. Their algorithm then places the streamlines in an order according to the scores. Based on the *linear entropy* proposed by Furuya and Itoh [5], which measures the complexity of a streamline according to the lengths of its line segments, the measurement of shape complexity proposed by Marchesin *et al.* also considers *angle entropy* derived from the angles of consecutive line segments. Besides, they count the number of streamlines overlapped with the streamline in question, and compute the average number of overlapped streamlines per occupied pixel. Their final metric uses the sum of linear entropy and angle entropy to measure the shape complexity, and divides the sum by the number of overlapped streamlines to penalize the streamlines in cluttered regions.

Figure 9 shows the streamlines with highest scores from the measurements proposed by Marchesin *et al.*, and our metric for the dataset *Plume*. Figure 9 (a-1) shows the streamlines with positive

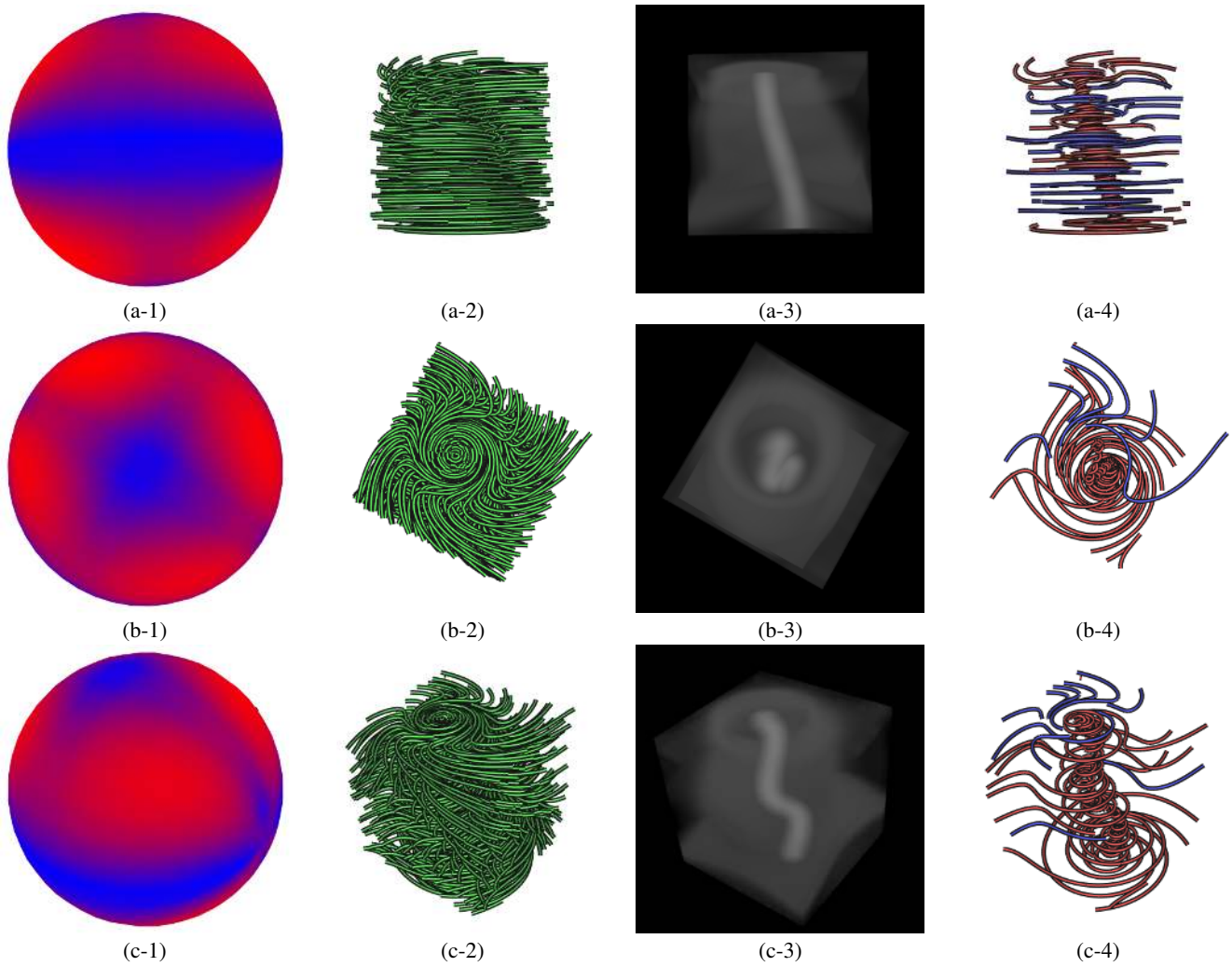


Figure 8: Viewpoint selection for *Tornado*. The figures from left to right, respectively, represent scores for the corresponding viewpoints, the vector field, the obtained MEP framebuffer, and the result from our view-dependent streamline placement algorithm. In the leftmost column, the center of the sphere's projection encodes the score for the corresponding viewpoint. Scores are mapped from blue (low) to red (high). Figures in the top two rows show two poor viewpoints. The top row shows one of the low scoring views. The tornado is viewed from one of the sides and thus the circular pattern of the flow field is not revealed. The middle row represents another low scoring view. The tornado is viewed from the top and thus the depth cue of the flow is not revealed. Bottom row: one of the good viewpoints.

scores by using our metric, which contains 251 streamlines. Figure 9 (b-1) shows the top 251 streamlines according to their shape complexity measure without being divided by the overlapping streamline count. It can be seen that the eddies inside the flow, features that are considered more important, are occluded. Figure 9 (c-1) shows the top 251 streamlines according to Marchesin *et al.*'s final metric. We can see that no streamlines near the eddies are selected. To explain this, our hypothesis is that their metric penalizes the streamlines in regions of high density, while the initial streamlines here were created by the algorithm proposed by Xu *et al.* [22] that places denser streamlines near more salient flow features. To verify our hypothesis, we tested the three metrics using another set of streamlines, which are randomly placed in space. By comparing Figures 9 (c-1) and (c-2), it can be seen that more streamlines near the eddies are selected by the metric proposed by Marchesin *et al.*, but the selected streamlines can still cause occlusion, while our evaluation prefers streamlines not only near the flow features but also cause less occlusion, as shown in Figures 9 (a-1) and (a-2).

Our streamline selection algorithm is also compared with the information-theoretic framework proposed by Xu *et al.* [22] since both methods try to control the placement of streamlines according to flow complexity, except that their method is view-independent and streamlines are placed according to object-space criteria. Due to object-based nature of their method, it may cause occlusion of regions of high complexity. For testing, we used streamlines generated by their algorithm as the input streamlines for our method. We compare visualization with n streamlines selected by our algorithm and visualization with the first n streamlines generated by their algorithm. Comparison of approaches is presented in Figure 10. We see that occlusion in Figure 10 (b) exists and hence it hinders a clear view of the more turbulent region. We also note that streamlines generated by their method do not provide a complete coverage of the flow in the dataset. With the same number of streamlines in both methods, our method does not display streamlines that cause occlusion. Therefore, we can produce a better coverage of the flow field for the given streamline count.

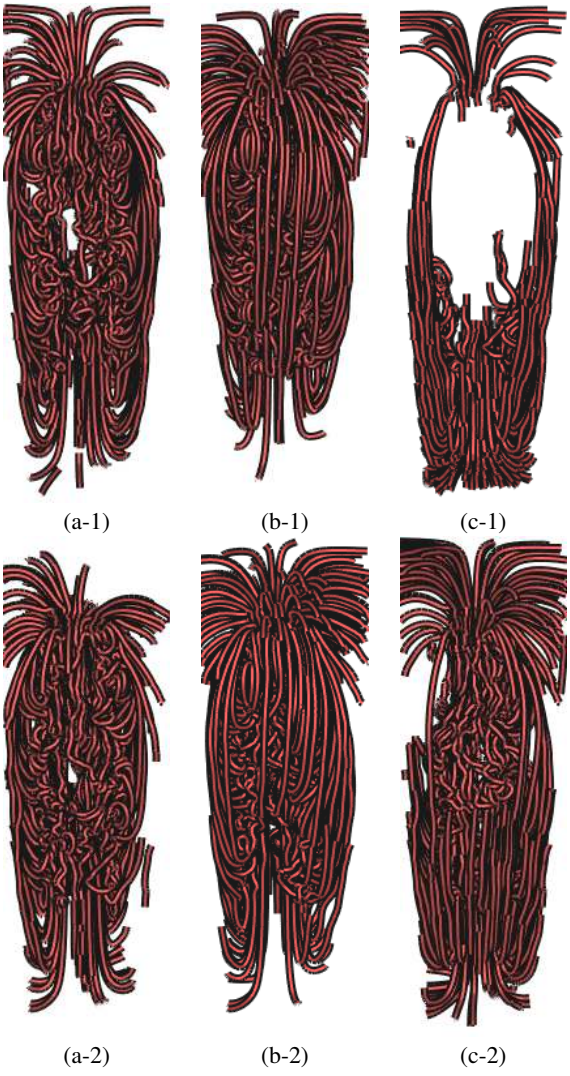


Figure 9: Comparison of our streamline evaluation scheme with the metrics proposed by Marchesin *et al.* [12] for *Plume*. The first row shows the results using the testing set of streamlines for other figures, while the bottom row uses another set of 1991 streamlines that are randomly placed. (a-1) and (a-2): The streamlines with positive scores from our streamline evaluation. 312 and 229 streamlines are rendered, respectively. The metrics used in (b) and (c) are the sum of linear entropy and angle entropy and the sum divided by the overlap, respectively. The screen resolution is 192×384.

Table 1: Test datasets and performance (milliseconds) under different screen sizes

Dataset	Tornado			Plume		
Size	$100 \times 100 \times 100$			$126 \times 126 \times 512$		
# Seeds	417			1991		
Screen Size	MEP	EVAL CPU/GPU	SEL	MEP	EVAL CPU/GPU	SEL
512^2	14	1234/6	821	32	5490/8	3362
768^2	33	1696/11	1721	49	6751/12	6286
1024^2	58	2099/18	3324	76	8041/18	10662

7 Performance

In this section we discuss the performance of our algorithm. We tested our implementation on a machine with Intel Core 2 Duo 6700 CPU with 2.66GHz, 3GB of RAM, and nVidia GeForce 460GTX GPU. Table 1 lists the timing of our streamline placement algorithm for test datasets with different screen resolutions, where the numbers represent average computation time measured from different viewpoints.

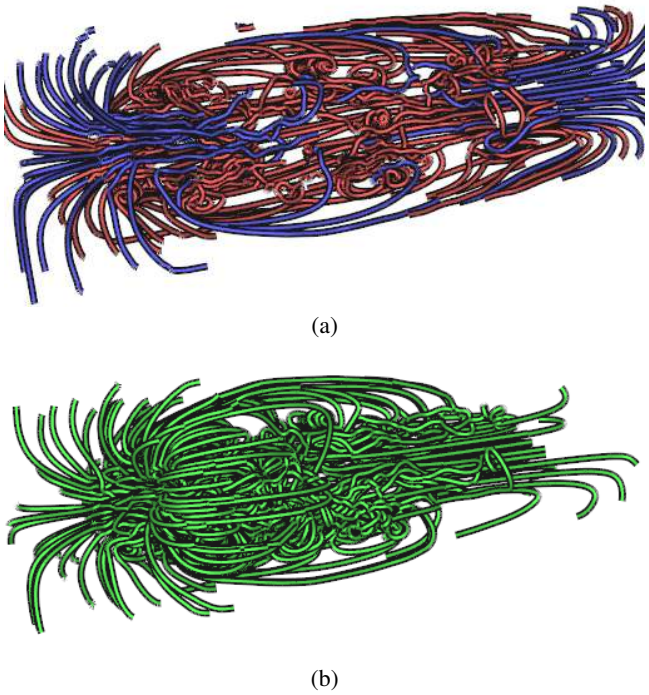


Figure 10: Comparison between our method (a) and the algorithm by Xu *et al.* [22] (b). In both cases 131 streamlines are used. Image (a) has size of 512×256 pixels and uses 64×64 tiles. Streamline width is $width = 4$.

The columns MEP in Table 1 show the time needed to construct the MEP framebuffer in milliseconds, which is achieved with a GLSL-based ray-caster on the GPU. We can see that the evaluation can be done interactively even for a large resolution such as 1024×1024. Consequently, our viewpoint selection algorithm is also fast since it essentially constructs MEP framebuffers from different viewpoints. For viewpoint selection, we set the resolution of the MEP framebuffer to 512×512 pixels, and it only took 24 seconds in total to evaluate 780 viewpoints for *Plume*.

The columns EVAL in Table 1 show the timings of streamline evaluation, using both CPU-based and GPU-based implementations. To utilize GPUs for streamline evaluation, we use nVidia CUDA. A straightforward implementation is to evaluate one streamline by a single CUDA thread, but the workload of the CUDA threads can be unbalanced as the streamlines can have different lengths. To address this, our implementation uses one CUDA thread to evaluate the score for one single streamline segment. Once the scores of the segments of all streamlines have been computed, we use the scan primitives proposed in [15] implemented in CUDPP, a CUDA-based utility library for GPGPU, to efficiently obtain the scores for all streamlines from their line segments in parallel. By comparing the timings of CPU and GPU implementations, we see that performance is accelerated by at least 100 times using GPUs.

Compared to streamline evaluation, accelerating streamline selection using GPUs is more complicated. The major reason is that the selection result depends on the order in which streamlines are examined, which will in turn influence the streamline densities of the image tiles in the intermediate steps. Consequently, the stream-

lines cannot be examined in parallel. The columns SEL in Table 1 show the timings of streamline selection. It is the major performance bottleneck of our algorithm and therefore further study is necessary to improve performance.

Regarding memory requirement, our algorithm has small memory footprint. For streamline placement, an array is needed to store the scores for all streamlines, and another array is required to store the scores for all viewpoints for the purpose of viewpoint selection, in addition to a 2D array for the MEP framebuffer. In our tests, the MEP framebuffer consumed most of the memory since its resolution was at least $512x512$ while less than 10000 streamlines and 780 viewpoints were tested. The required memory space, nevertheless, for a 1024×1024 MEP frambuffer in single precision was only 4MB, which is not an issue for moderate personal computers.

8 Conclusion and Future Work

In this paper we present algorithms that minimize occlusion for geometry-based vector field visualization using an information-theoretic approach. Our algorithms measure the local complexity within a vector field using entropy and store the entropy as a three-dimensional volume. It is then used to evaluate a Maximal Entropy Projection (MEP) framebuffer which utilizes the entropy in a view-dependent manner. The MEP framebuffer stores the maximal entropy values and corresponding z-values for a given view, allowing our algorithms to select streamlines or viewpoints based on the occlusion and visibility of salient flow features. We demonstrate the benefits of our algorithms with a number of test cases and compare our results with related approaches.

For future work, we plan to perform a user study to get feedback from domain scientists about the usefulness and quality of our approach. We also plan to extend the idea of view-dependent flow complexity to generate view-independent streamlines. Since our algorithm is view-dependent, the coherence between different viewpoints for streamline placement needs to be considered, otherwise potential popping artifacts can occur when changing the viewpoint. We will consider to adapt our viewpoint selection algorithm to work with other flow rendering methods. For instance, while we currently render streamlines as opaque polylines, using transparency might be helpful to reveal the overlapped flow features, requiring the revision of our MEP-framebuffer-based metric. Finally, we would like to extend our algorithm to handle time-varying vector fields.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments. This work was supported in part by NSF grant IIS-1017635, US Department of Energy DOE-SC0005036, Battelle Contract No. 137365, Los Alamos National Laboratory Contract No. 69552-001-08, and Department of Energy SciDAC grant DE-FC02-06ER25779. The dataset *Plume* was released by the National Center for Atmospheric Research.

References

[1] U. Bordoloi and H.-W. Shen. View selection for volume rendering. In *Vis '05: Proceedings of the IEEE Visualization 2005*, pages 487–494, 2005.

[2] M. Chen and H. Jänicke. An information-theoretic framework for flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1206–1215, 2010.

[3] Y. Chen, J. Cohen, and J. Krolik. Similarity-guided streamline placement with error evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1448–1455, 2007.

[4] R. A. Crawfis and N. Max. Texture splats for 3d scalar and vector field visualization. In *Vis '93: Proceedings of the IEEE Visualization '93*, pages 261–266, 1993.

[5] S. Furuya and T. Itoh. A streamline selection technique for integrated scalar and vector visualization. In *Vis '08: IEEE Visualization 2008, Poster Session*, 2008.

[6] H. Hauser, R. S. Laramee, and H. Doleisch. State-of-the-art report 2002 in flow visualization. 2002.

[7] G. Ji and H.-W. Shen. Dynamic view selection for time-varying volumes. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1109–1116, 2006.

[8] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. In *Proceedings of Eighth Eurographics Workshop on Visualization in Scientific Computing*, pages 45–55, 1997.

[9] P. Kohlmann, S. Bruckner, A. Kanitsar, and E. Gröller. Livesync: Deformed viewing spheres for knowledge-based navigation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1544–1551, 2007.

[10] R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf. The state of the art in flow visualization: Dense and texture-based techniques. *Computer Graphics Forum*, 23(2):203–221, 2004.

[11] L. Li and H.-W. Shen. Image-based streamline generation and rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):630–640, 2007.

[12] S. Marchesin, C.-K. Chen, C. Ho, and K.-L. Ma. View-dependent streamlines for 3d vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1578–1586, 2010.

[13] T. McLoughlin, R. Laramee, R. Peikertand, F. Post, and M. Chen. Over two decades of integration-based, geometric flow visualization. *Computer Graphics Forum*, 29(6):1807–1829, 2010.

[14] A. Mebarki, P. Alliez, and O. Devillers. Farthest point seeding for efficient placement of streamlines. In *Vis '05: Proceedings of the IEEE Visualization 2005*, pages 479–486, 2005.

[15] S. Sengupta, M. Harris, Y. Zhang, and J. D. Owens. Scan primitives for gpu computing. In *Graphics Hardware 2007*, pages 97–106, 2007.

[16] B. Spencer, R. S. Laramee, G. Chen, and E. Zhang. Evenly spaced streamlines for surfaces: An image-based approach. *Computer Graphics Forum*, 28(6):1618–1631, 2009.

[17] S. Takahashi, I. Fujishiro, Y. Takeshima, and T. Nishita. A feature-driven approach to locating optimal viewpoints for volume visualization. In *Vis '05: Proceedings of the IEEE Visualization 2005*, volume 0, pages 495–502, 2005.

[18] G. Turk and D. Banks. Image-guided streamline placement. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 453–460, 1996.

[19] P.-P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich. Automatic view selection using viewpoint entropy and its applications to image-based modelling. *Computer Graphics Forum*, 22(4):689–700, 2003.

[20] V. Verma, D. T. Kao, and A. Pang. A flow-guided streamline seeding strategy. In *Vis '00: Proceedings of the IEEE Visualization 2000*, pages 163–170, 2000.

[21] I. Viola, M. Feixas, M. Sbert, and M. E. Gröller. Importance-driven focus of attention. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):933–940, 2006.

[22] L. Xu, T.-Y. Lee, and H.-W. Shen. An information-theoretic framework for flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1216–1224, 2010.

[23] X. Ye, D. T. Kao, and A. Pang. Strategy for seeding 3D streamlines. In *Vis '05: Proceedings of the IEEE Visualization 2005*, pages 471–478, 2005.