Illustrative Streamline Placement and Visualization

Liya Li*
The Ohio State University

Hsien-Hsi Hsieh†
National Dong Hua University, Taiwan

Han-Wei Shen‡
The Ohio State University

ABSTRACT

Inspired by the abstracting, focusing and explanatory qualities of diagram drawing in art, in this paper we propose a novel seeding strategy to generate representative and illustrative streamlines in 2D vector fields to enforce visual clarity and evidence. A particular focus of our algorithm is to depict the underlying flow patterns effectively and succinctly with a minimum set of streamlines. To achieve this goal, 2D distance fields are generated to encode the distances from each grid point in the field to the nearby streamlines. A local metric is derived to measure the dissimilarity between the vectors from the original field and an approximate field computed from the distance fields. A global metric is used to measure the dissimilarity between streamlines based on the local errors to decide whether to drop a new seed at a local point. This process is iterated to generate streamlines until no more streamlines can be found that are dissimilar to the existing ones. We present examples of images generated from our algorithm and report results from qualitative analysis and user studies.

1 INTRODUCTION

Effective visualization of vector fields plays an important role in many scientific and engineering disciplines. To visualize 2D flow fields, several techniques have been developed in the past. The better known techniques include the geometry-based methods such as streamline and particle tracing, and the texture-based methods such as LIC [2], Spot Noise [15], and IBFV [16]. In general, texture-based techniques display a dense representation of the flow fields but the resulting visualization generally lacks visual focus to highlight salient flow features. They are also more expensive to compute and can suffer from aliasing problems. Comparatively speaking, visualization of streamlines is still a popular method because they are faster to compute and can be rendered at any resolution at interactive rates. The main challenge for the streamline-based methods, however, is the placement of seeds. On the one hand, placing too many streamlines can make the final images cluttered, and hence the data become more difficult to understand. On the other hand, placing too few streamlines can miss important flow features. An ideal streamline seed placement algorithm should be able to generate visually pleasing and technically illustrative images. It should also allow the user to focus on important local features in the flow field.

Hand-drawn streamlines are frequently shown in scientific literature to provide concise and illustrative descriptions of the underlying physics. Fig. 1 shows such an example. The abstract information provided by the streamlines in the image clearly shows the primary features of the flow field. Even though the streamlines do not cover everywhere in the field, we are able to create a mental model to reconstruct the flow field when looking at this concise illustration. That means, to depict a flow field, it is unnecessary to draw streamlines at a very high density. Abstraction can effectively prevent visual overload, and only emphasizes the essential while deemphasizing the trivial or repetitive flow patterns. In the visualization research literature, there have been some streamline seeding algorithms proposed in the past [14, 6, 17, 10, 9]. Most of the methods, however, are based on evenly-spaced distribution criteria, i.e., streamlines are spaced evenly apart at a pre-set distance threshold across the entire field. While those methods can reduce visual cluttering by terminating the advection of streamlines when they are too close to each other, more streamlines than necessary are often generated as a result. In addition, there is no visual focus provided to the viewers to quickly identify the overall structure of the flow field.

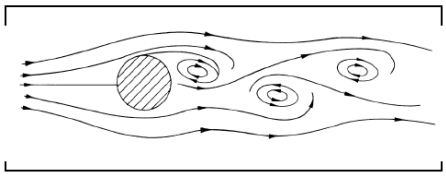


Figure 1: Hand-drawn streamlines for a flow field around a cylinder. Image courtesy of Greg Turk [14].

Spatial coherence often exists in a flow field, meaning neighboring regions have similar vector directions, and nearby streamlines resemble each other. To create a concise and illustrative visualization of streamlines, in this paper we present a seeding strategy which utilizes spatial coherence of streamlines in 2D vector fields. Our goal is to succinctly and effectively illustrate vector fields, rather than uniformly laying out streamlines with equal distances between them, as in most of the existing methods. We allow the density of streamlines in the final image to vary to reflect the coherence of the underlying flow patterns and also to provide visual focus. In our algorithm, 2D distance fields representing the distances from each grid point in the field to the nearby streamlines are computed. From the distance fields, a local metric is derived to measure the dissimilarity between the vectors from the original field and an approximate field computed from the nearby streamlines. We also define a global metric to measure the dissimilarity between streamlines. A greedy method is used to choose a point as the next seed if both of its local and global dissimilarity satisfy our requirements.

The remainder of the paper is organized as follows. We first briefly review the related work. We then present our illustrative streamline placement algorithm in detail. Finally, results from quantitative analysis and user studies are discussed.

2 RELATED WORK

There exist several streamline seeding strategies for two dimensional flow fields. The image guided streamline placement algorithm proposed in [14] uses an energy function to measure the difference between a low-pass filtered streamline image and an image of the desired visual density. An iterative process is used to reduce the energy through some pre-defined operations on the streamlines. Jobard and Lefer [6] explicitly control the distance between adjacent streamlines to achieve the desired density. Each candidate seed is placed at a point away from an existing streamline at

*e-mail: lil@cse.ohio-state.edu
†e-mail:hsi@game.csie.ndhu.edu.tw
‡e-mail:hwshen@cse.ohio-state.edu

some pre-specified distance, from which a new streamline is advected backward and forward until it comes too close to the existing streamlines or leaves the 2D domain. Verma et al. [17] proposed a seed placement strategy based on flow topology characterized by critical points in the field. Different seeding templates for various types of critical points are defined, and the shape and size of templates are determined by the influence region covered by the critical points. To have a sufficient coverage, additional seed points are randomly distributed in empty regions using Poisson disk distribution. Mebarki et al. [10] proposed a two dimensional streamline seeding algorithm by placing a new streamline at the farthest point away from all existing streamlines. Delaunay triangulation is used to tessellate regions between streamlines, and seeds are placed in the center of the biggest voids. The purpose of their algorithm is to generate long and evenly spaced streamlines. Intuitively, seeding a streamline in the largest empty region indeed favors longer streamlines. However, as more streamlines are generated or a smaller separating distance threshold is used, especially near the critical regions, discontinuity of flow paths can still exist. Liu et al. [9] proposed an advanced evenly-spaced streamline placement strategy which prioritizes topological seeding and long streamlines to minimize discontinuities. Adaptive distance control based on local flow variance is used to address the cavity problem.

There are some other research related to our work. In the past years, there are many techniques proposed to simplify vector fields. Heckel et al. [5] proposed to generate a top-down segmentation of the discrete field by splitting clusters of points. Beginning with a single cluster including all points of the vector field, more clusters are created under the guidance of some error metric defined by the distance between the streamline integrated from the simplified vector field and one from the original vector field. Telea and Wijk [11] presented a method to hierarchically bottom-up cluster the input flow field. The algorithm repeatedly selects the two most resembling neighboring clusters and merging them to form a larger cluster until a single cluster covering the whole field is generated. The metric to evaluate the similarity between vectors is based on the direction and magnitude comparison, and the position comparison. Du and Wang [3] proposed to use Centroidal Voronoi Tessellations (CVTs) to simplify and visualize the vector fields. Chen et al. [**?**] extracted the topology of a vector field using Morse Decomposition, which can be used to decide the locations of streamlines. In terms of measuring the similarity between streamlines, Bordoloi and Shen [1] presented an interactive global technique for dense vector field visualization using levels of detail. The level of detail is controlled based on the local complexity of the vector field. A quadtree is constructed and used as a hierarchical data structure for error measurement. The error associated with each node represents the error when only one representative streamline is computed for all the points within the entire region corresponding to the node.

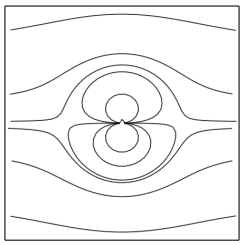


Figure 2: Streamlines generated by our algorithm.

3 Algorithm Overview

The primary goal of our work is to generate streamlines succinctly for 2D flow fields by emphasizing the essential and deemphasizing the trivial or repetitive flow patterns. Fig. 2 shows an example of streamlines generated by our algorithm, where the selection of streamlines is based on a similarity measure among streamlines in the nearby region. The similarity is measured locally by the directional difference between the original vector at each grid point and an approximate vector derived from the nearby streamlines, and globally by the accumulation of the local dissimilarity at every integrated point along the streamline path. To approximate the vector field from the existing streamlines, 2D distance fields recording the closest distances from each grid point in the field to the nearby streamlines are first computed. Then the approximate vector direction is derived from the gradients of the distance fields. Our algorithm greedily chooses the next candidate seed that has the least degree of similarity according to our metrics. Detailed information about our similarity measures and the seed selection algorithm is provided in later sections.

Our algorithm has the following unique characteristics when compared with the existing streamline seeding algorithms [14, 6, 17, 10, 9]:

First, the density of streamlines. Some of the existing techniques favor uniformly spaced streamlines. However, in our algorithm, the density of streamlines is allowed to vary in different regions. The different streamline densities reflect different degrees of coherence in the field, which allows the viewer to focus on more important flow features. Regions with sparse streamlines imply the flows are relatively coherent, while regions with dense streamlines mean more seeds are needed to capture the essential flow features. This characteristic of our algorithm matches with one of the general principles of visual design by Tufte [13] - different regions should carry different weights, depending on their importance. The information can be conveyed in a layered manner by means of distinctions in shape, color, density, or size.

Second, the representativeness of streamlines. The general goal of streamline placement is to visualize the flow field without missing important features, which can be characterized by critical points. Since the flow directions around critical points can change rapidly compared to those non-critical regions, our algorithm is able to capture those regions and place more streamline seeds accordingly.

Finally, the completeness of flow patterns. In the previous streamline placement algorithms that have explicit inter-streamline distance control, the advection of streamlines can be artificially terminated. This may cause visual discontinuity of flow pattern, especially when it is near the vicinity of critical points. Our seeding algorithm, however, only determines where to drop seeds and allows the streamlines to be integrated as long as possible until they leave the 2D domain, reach critical points, or generate a loop. Without abruptly stopping the streamlines, the flow patterns shown in the visualization are much more complete and hence easier to understand.

In the following, we discuss our algorithm in detail.

3.1 Distance Field

A distance field [7] represents the distance from every point in the domain to the closest point on any object. The distance can be unsigned or signed, and the sign is used to denote that the point in question is inside or outside of the object. With the distance field, some geometric properties can be derived such as the surface normal [4]. The concept of distance fields has been used in various applications such as morphology, visualization, modeling, animation, and collision detection.

In our algorithm, we use unsigned distance fields to record the closest distance from every point in the field to the nearby streamlines that have been computed. In practice, a mathematically smooth streamline is approximated by a series of polylines integrated bidirectionally through numerical integrations. Given

a line segment $s_i = \{p_i, p_{i+1}\}$, where $p \in R^3, i \in N$, a vector $v_i = p_{i+1} - p_i$, we can compute the nearest point p_q on the line segment s_i to an arbitrary point q by:

$$p_q = p_i + t v_i \tag{1}$$

where

$$t = clamp(\frac{(q - p_i) \cdot v_i}{|v_i|^2}),$$
$$clamp(x) = min(max(x, 0), 1), \tag{2}$$

The distance $d(q, s_i)$ from the point q to the line segment s_i is computed by the Euclidean distance between q and p_q. For a given streamline L, where $L = \{\cup s_i | s_i = \{p_i, p_{i+1}\}, i \in N, p \in R^3\}$, and s_i is a line segment of line L, the unsigned distance function at a point q with respect to L is:

$$d(q, L) = \min\{d(q, s_i) | s_i \in L\} \tag{3}$$

To speed up the computation of distance fields, we implement it on GPU. We defer the discussion about the GPU implementation to section 6. The distance fields are used to derive an approximate vector field, which can be used to measure the dissimilarity between streamlines in the local regions. In the next section, we describe our algorithm in detail.

3.2 Computation of Local Dissimilarity

Because of spatial coherence in the field, neighboring points can have advection paths with similar shapes, even though they may not be exactly the same. Given a streamline, considering the closest distance from every point in the field to this streamline, a distance field can be computed. The iso-contours of this distance field will locally resemble the streamline, i.e., the closer is the contour to the streamline, the more similar their shapes will be. This is the basic idea how we locally approximate streamlines in the empty regions from existing ones, which forms the basis for us to measure the coherence of the vector field in local regions.

With the the distance field, we compute a gradient field using the central difference operator. For each vector of this gradient field, after a 90 degree of rotation, we get an approximate vector that is derived from the single streamline. Whether to rotate the gradient clockwise or counter-clockwise is decided based on the flow direction of the nearby streamline so that the resulting approximate vector points to roughly the same direction as the streamline. To measure the local coherence, we define a local dissimilarity metric as the direction difference between the true vector at the point in question and its approximate vector. For a point $p \in R^3$, the local dissimilarity $D_l(p)$ at this point is written as the following:

$$D_l(p) = 1 - (\frac{v\prime(p) \cdot v(p)}{|v\prime(p)||v_(p)|} + 1)/2 \tag{4}$$

where $v\prime(p)$ is the approximate vector at p, and $v(p)$ the original vector. The value is in the range of 0.0 to 1.0; the larger the value is, the more dissimilar between the true vector and the approximate vector at that point. We note that this metric only denotes the local dissimilarity between the vectors at the point, instead of the dissimilarity between the streamline originated from this point and its nearby streamline. Also, so far we only consider the case that there exists only one streamline in the field. In the next section, we discuss how to consider multiple streamlines existing in the field and modify our dissimilarity metric, which is a more general case assumed in our algorithm. After that, we discuss how to select the streamline seeds.

3.3 Influence from Multiple Streamlines

When there exist multiple streamlines in the field, we cannot use the standard definition of distance field and simply compute one smallest distance from each point to the streamlines, and evaluate the dissimilarity metric as presented above. This is because the distance field computed with this method will generate a discrete segmentation of the field. For example, the left image in Fig. 3 shows the approximate vectors in orange given two existing streamlines S_1 and S_2 in black. For the points in the lower triangular region under the dotted line, they are classified to be the closest to streamline S_2, while the points in the upper triangle are the closest to streamline S_1. If we use a single distance field computed from the two lines to approximate the local vectors, the resulting vectors will be generated in a binary manner, as shown by those orange vectors. This binary segmentation causes discontinuity in the approximate vector field. Given two lines as shown in the example, for the empty space in between, a more reasonable approximation of the vectors should go through a smooth transition from one line to the other, as shown on the right in Fig. 3.

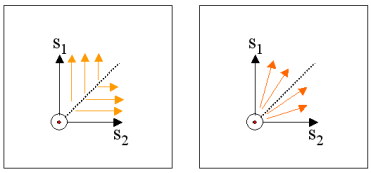


Figure 3: Assume the flow field is linear and streamlines are straight lines. The circle in the images denotes the region where a critical point is located. Black lines represent the exact streamlines seeded around the critical point. The orange lines represent the approximate vectors by considering the influence of only one closest streamline (left), and the blending influence of two closest streamlines (right).

In our algorithm, we achieve a smooth transition of vector directions between streamlines by blending the influences from multiple nearby streamlines. In the previous section, we discuss how to compute the dissimilarity metric if there exists only one streamline. For the more general case where multiple streamlines are present, for each point we pick the M nearest streamlines, evaluate the dissimilarity function as in equation 4 for each streamline respectively, and blend the M $D_{lk}(p)$ together to compute the final dissimilarity value at p as:

$$D_l(p) = \sum_{k=1}^{M} (w_k D_{lk}(p)) \tag{5}$$

where w_k is the weight of the influence from the streamline k decided by the distance between point p and the streamline k. $D_{lk}(p)$ is the dissimilarity value computed at point p using the distance field generated by streamline k. Analogously, the approximate vector at p is the blending of the vectors generated from the M nearest streamlines, and each vector is a 90 degree rotation of the gradient computed from the corresponding streamline, as described above. We note that different methods for assigning the weight can be used in the equation depending on the requirement of the user. For all the images presented in this paper, we consider the blending of two nearest streamlines, that is, M equals to 2 in equation 5.

3.4 Computation of Global Dissimilarity

As mentioned in the previous section, at each point, there is a local dissimilarity measure that represents the direction difference between the true vector at that point and the approximate vector derived from the nearby streamlines. However, the local dissimilarity only captures the coherence about the local vectors instead of the similarity between streamlines. In order to capture the coherence

between a streamline originated from a point and its nearby streamlines, we define a global dissimilarity measure by accumulating the local dissimilarity at every integrated point along its streamline path. Written in equation:

$$D_g(p) = \sum_{n=1}^{L} (u_n D_l(x_n, y_n)) \quad (6)$$

where $D_g(p)$ is the global dissimilarity at point p, and (x_n, y_n) is the nth integrated point along the streamline originated from p. The length of the streamline is L. $D_l(x_n, y_n)$ is computed by interpolating the local dissimilar values at the four corner grid points. Based on different metrics, u_n can be computed differently. In our algorithm, we use averaged local dissimilarity values along the streamline path, i.e., u_n is equal to $1/L$.

3.5 Selection of Candidate Seeds

Before we discuss our algorithm, we first introduce two user-specified threshold values, T_l and T_g. T_l is the threshold for the minimum local dissimilarity, while T_g is the threshold for the minimum global dissimilarity. To avoid drawing unnecessary streamlines, we only choose seeds from grid points satisfying equation 7.

$$D_l(i, j) > T_l; D_g(i, j) > T_g \quad (7)$$

The initial input to our algorithm is a streamline seeded at a random location in the field. For example, we can use the central point of the domain as the initial seed to generate the streamline. With the first streamline, the distance field is calculated and the dissimilarity value at each grid point is computed. The important step now is how to choose the next seed. Here we present a greedy but more efficient method for this purpose. Given the two threshold values, our algorithm for choosing the next seed is described as follow:

1. Sort the grid points in the descending order of the local dissimilar values computed from equation 5.

2. Dequeue the first point (i, j) in the sorted queue. If $D_l(i, j)$ is larger than T_l, integrate a streamline from this point bidirectionally and compute the global dissimilarity value $D_g(i, j)$ by using equation 6. Otherwise, if $D_l(i, j)$ is smaller than T_l, the iteration terminates.

3. If $D_g(i, j)$ is larger than T_g, this seed is accepted as the new seed and the streamline being integrated is displayed. Otherwise, go back to step (2).

When a new streamline is generated, we update the distance field and re-compute the dissimilarity values at the grid points as mentioned in section 3.3. The above algorithm runs iteratively to place more streamlines. The more streamlines we place, the smaller the dissimilarity values will become at the grid points. The program terminates when no seed can be found that satisfies equation 7. At this point, we have enough streamlines to represent the underlying flow field according to the user desired coherence thresholds.

To speed up the process of choosing the candidate seeds, during the process mentioned above, when $D_g(i, j)$ is smaller than T_g, we mark this grid point, and also those grid points at the four corners of the cells that are passed by the streamline originated from (i, j). These points will be excluded from being considered any further in the later iterations, because there already exist nearby streamlines very similar to the streamlines that would have been computed from them. Therefore, it is unnecessary to check those grid points again. Generally speaking, for a dataset that has a sufficient resolution, the flow within a cell is very likely to be coherent, so this heuristic will not affect the quality of our visualization output much. That means, in most cases, streamlines from those grid points will be similar to the streamline that has already been rejected. This allows us to reduce the number of streamlines to compute and test substantially, without visible quality difference being seen from all of our experiments.

Fig. 4 shows an image of streamlines generated with an ocean-field data using our algorithm. For rendering, since our algorithm allows streamlines to be integrated as long as possible until they leave the 2D domain, reach critical points, or generate a loop, the local density of ink in some regions may be higher than other regions. To even the distribution of ink, we render the streamlines in the alpha blending mode, where the alpha value of each line segment is adjusted according to the density distribution of the projected streamline points in image space. Each sampling point on the streamlines is first mapped to image space, and the corresponding screen space point is treated as some energy source, which can be defined by the Gaussian function. Then, an energy distribution map based on all streamlines is generated. This energy map is mapped to an opacity map to control the opacity of the streamline line segments as they are drawn. This can effectively reduce the intensity of the lines if they are cluttered together.

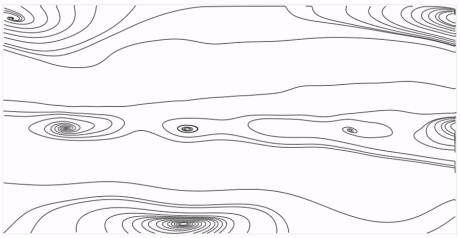


Figure 4: Streamlines generated by our algorithm on the Oceanfield data.

4 Topology-Based Enhancement

Although without explicitly considering the flow topology, our algorithm would naturally place more streamline seeds around the critical points because of the lack of coherence there. Sometimes it is desired to highlight the streamline patterns around the critical points so that the viewer can clearly identify the type of the critical points. To achieve this goal, we can adapt our algorithm by placing an initial set of streamlines with some specific patterns around the critical points, instead of randomly dropping the first seed. This is similar to the idea of seed templates proposed by Verma et al. [17]. For each type of critical points, we use a minimal set of streamlines to distinguish them from each other. For a source or sink, we place four seeds along the perimeter of a circle around the critical point, where each of the seeds is the intersection point of the x-y axes with the circle; for saddle, we place four seeds along the two lines bisecting the eigen directions with two seeds on each line; for spiral or center, we place one seed along a straight line emanating from the critical point. Fig. 5 shows such an image of streamlines generated with topology information being considered.

We note that streamline placement guided by topology information alone is not always effective, which can happen when there is no critical point, or there are too many critical points in the field. When there are too many critical points, the final image may easily get cluttered. On the other hand, if there is no critical point at all in the field, then no rules can be applied to guide the placement of streamlines. Our algorithm can consider both the vector coherence and the flow topology.

5 Quality Analysis

As mentioned above, our algorithm generates representative streamlines to illustrate the flow patterns of the underlying field. Given appropriate threshold values, our algorithm selects streamlines based on the flow coherence via the dissimilarity measures

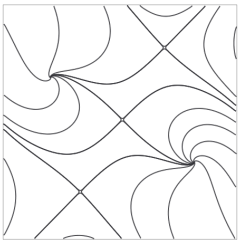


Figure 5: Streamlines generated when the flow topology is considered. There are three saddle and two attracting focus critical points in this data.

defined above. The density of the selected streamlines can vary based on the degree of coherence in the local regions. As in Fig. 4, there are void regions between the displayed streamlines, which tell us the streamlines in those void regions look similar to each other and hence can be easily derived. Therefore, our algorithm does not place many seeds in those regions. Since we only draw a small subset of the streamlines in the whole vector field, it is necessary to conduct quality analysis of our method. One method of analysis, which can be performed quantitatively, is to compare the original vector field with the approximate vector field derived from the streamlines selected by our algorithm. Another method is to perform user studies to verify whether the users can correctly interpret the field in the empty regions, and also whether our representation is an effective method to depict the vector fields. In the following, we first describe our approach for performing quantitative analysis with some results, and then present findings from our user studies.

5.1 Quantitative Comparison

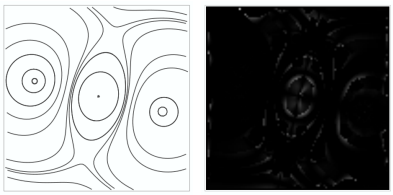
Our quantitative analysis consists of a data level comparison and a streamline level comparison. For the data level comparison, we first reconstruct a vector field from the streamlines generated by our algorithm. Then we compare the local vectors between the reconstructed field and the original field. For the streamline level comparison, originated from each grid point, two streamlines are integrated respectively in the original vector field and the reconstructed one, and we compute the errors between these two streamlines. We note that the errors are only used to study whether our algorithm misses any regions that require more streamlines to be drawn. The errors *do not* represent the errors in the visualization, since every streamline presented to the user is computed using the original vector field. In the following, we first describe how we reconstruct a vector field from the streamlines that are displayed. We then present our data level and streamline level comparison results.

5.1.1 Reconstruction of Flow Field

The process to reconstruct the approximate flow field from selected streamlines is very similar to the process presented in section 3.2 and 3.3 that we use to iteratively introduce streamline seeds. The main difference is that now we are given a final set of streamlines to generate the gradient fields. Given a streamline in the final streamline set, a distance field can be computed, from which we can compute its derived gradients. In section 3.3, we discuss the computation of the local dissimilarity by considering multiple nearby streamlines. With the same idea, for each grid point, we first identify the nearest M streamlines, and use the distances to the streamlines to generate M gradients at that point. After rotating the gradients by 90 degrees to get the approximate vectors, the final reconstructed vector at this grid point is computed from an interpolation of the M vectors inversely proportional to the distances from the point to the corresponding streamline. As mentioned above, in this paper we consider the nearest two streamlines for each grid point, that is, $M = 2$. For the grid points that are selected as the seeds or there are streamlines passing through it, we use the original vectors as the reconstructed vectors.

5.1.2 Data Level Comparison

Data level comparison is performed between the original vector field and the reconstructed vector field at every grid point. Our goal is to evaluate how well the streamlines displayed by our algorithm can represent the original vectors at the empty regions, based on the computational model we introduce above. One of the challenges to perform data level comparison is to design appropriate metrics to quantify the errors. Since our goal is to evaluate how much the true vector direction at each grid point is aligned with the reconstructed vector, we take the cosine of the angle between the original vector and the reconstructed vector at each grid point as a measure of similarity. Fig. 6 shows a result of our comparison using one vector data set. In the image, dark pixels depict that the two vectors at the grid points are almost the same, while brighter pixels mean more errors. From the image, it can be seen that the streamlines we display are representative for the original vector field, because in most of the empty regions, the approximate vectors from the streamlines are well aligned with those in the original field. There are a few regions with higher errors, which mostly fall into the following cases. The first case is regions near the domain boundary. Our algorithm explicitly excludes the grid points on the boundary from being selected as candidate seeds. This is because sometimes the vectors on boundaries are odd due to sampling issues, but the fieldlines in downstream or upstream tend to be more normal and stable. The second case of error is due to our implementation. When we select the next candidate seed, if a grid point is too near to an existing streamline, for example the distance to this streamline is within a cell, we exclude this point from being a candidate seed. This is really not a cause of concern because even if the streamline integrated from this point eventually will be different from this existing streamline, there will be some point elsewhere on this streamline or near this streamline being picked up as the seed. The third case might be a problem caused by the linear interpolation operator we use to blend the influence from multiple nearby streamlines based on the distance from the grid points to those streamlines.



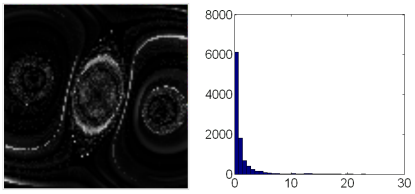
(a) (b)

Figure 6: (a) Representative streamlines generated by our algorithm (b) Gray scale image colored by one minus a normalized value of the cosine of the angle between vectors from the original field and the reconstructed field. Dark color means the two vectors are almost aligned with each other, while brighter color means more errors. The maximal difference between the vector directions in this image is about 26 degree, and the minimal difference is 0 degree.

5.1.3 Streamline Level Comparison

Besides comparing the original and the reconstructed vector fields with the raw data, we can also compare these two fields in terms of some global features, such as streamlines. To do this, from every grid point, we simultaneously integrate streamlines forward and backward in the original vector field and the reconstructed field, and then compute the distance between those two streamlines at every

integration step based on some metrics, such as Euclidean distance, or Manhattan distance. Fig. 7 shows a result of streamline comparison on the same vector fields as Fig. 6, where we compute the average Euclidean distance between the two streamlines. Similar to the cases discussed in section 5.1.2, some errors are detectable in some local regions but they are quite small. In Fig. 7 (b), we show the histogram of the distance errors, from which we can see that most of the grid points from which the streamlines originated only bear small errors.



(a) (b)

Figure 7: (a) Gray scale image colored by the distance errors (in the unit of cells) between two streamlines integrated from each grid point in the original vector field and the reconstructed one. Dark color means low errors, while brighter color means higher errors (b) Histogram of the streamline errors collected from all grid points in the field. X axis is the error, while Y axis is the frequency of the corresponding error value. The maximal difference is 23.1 and the minimal is 0.0. The dimensions of the field is 100 by 100.

5.2 User Study

Abstract or illustrative presentations have been widely used and accepted in non-photorealistic rendering and artistic design to depict information succinctly. User study is a way to quantify the effectiveness of new methods, like in [8]. To evaluate the effectiveness of using illustrative streamlines generated by our algorithm, we conducted a user study which contained four questions categorized into two tasks. The tasks and questions were related to visualization of four different 2D vector fields. In the following, we describe our study and discuss the results.

5.2.1 Participants

Subjects for the user study were 12 unpaid graduate students from the Department of Computer Science and Engineering. Five of them are majored or will be majored in Computer Graphics, and others are in other research groups, such as Artificial Intelligence, Networking, etc. Two of them know a little about the concept of flow fields and streamlines, but none of them had studied fluid mechanics or related courses. There were four female students and eight male students. They all have normal or corrected visions and can see the images presented to them clearly. The study took about 30 mins for each subject, and before the test, the subjects were given a tutorial introducing them to the application. We explained the purpose of using streamlines to visualize flow fields, and different flow features being depicted by different types of critical points. The tests did not start until they could easily tell the flow features in the training datasets without our help.

5.2.2 Tasks and Procedure

Our first task was to evaluate whether the users were able to effectively identify the underlying flow features, including flow paths and critical points, from the visualization generated by our algorithm. In particular, we wanted to verify whether our streamline representation was as effective as other existing algorithms, or more, in terms of allowing the users to understand the vector fields. This part was conducted on pieces of paper handed out to the subjects and there were three questions involved.

To perform the test, we chose two existing 2D streamline placement algorithms by Mebarki et al. [10] and Liu et al. [9], plus our method, and generated images using four datasets. The subjects were shown 15 groups of images, and each group included three images generated by the three algorithms respectively. For the images within each group generated by the algorithms of Mebarki and Liu, the streamline densities were similar, but between different groups, the density of streamlines were different. To avoid possible bias caused by a fixed ordering of images by the three algorithms, we changed the order of three images randomly in each group. Fig. 8 shows three groups of images used in our user study. At the beginning of this task, instructions were given to the subjects about the questions in detail. They were required to fully understand the questions before they started to give answers.

The first question in the test was to ask the subjects to rate the three images in each group according to the easiness of depicting the flow paths in the vector fields, where 1 was the best and 3 was the worst. The second question was about critical points. If there were critical points in the fields, subjects were asked to circle them and rate how helpful the streamlines presented in the visualization were to detect those critical points. The third questions was about the overall effectiveness of visualization considering both the flow paths and critical points.

In the study, we did not ask the subjects to classify the critical points. If the subjects thought all three images were equally helpful, then they could rate them equally.

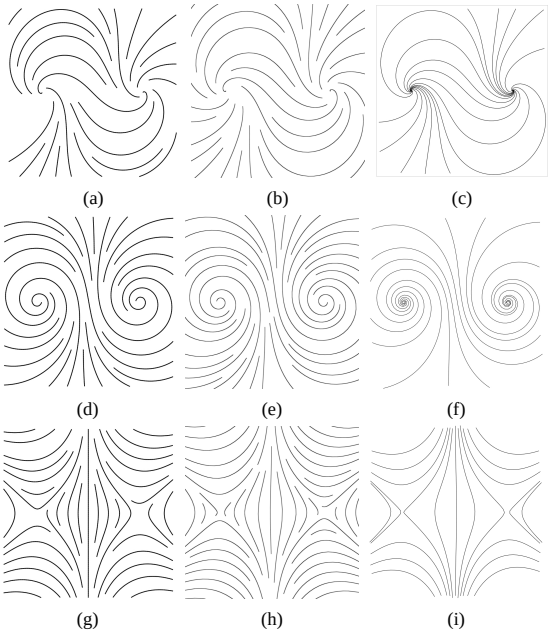


Figure 8: Streamlines generated by Mebarki et al. 's algorithm (left), Liu et al. 's algorithm (middle), and our algorithm (right).

Our second task was to evaluate how correctly the subjects were able to interpret the flow directions with the images generated from our algorithm in those empty regions without streamlines being drawn. This task was run with a completely automated program with four datasets. We pre-generated streamlines using our algorithm on each data set, which were used as the input to the program. When the program started with each data set, four random seed points were generated in those void regions. For each point, six circles with increasing radii were generated in a sequence. The subjects were asked to mark where the streamlines would intersect with the circles when they were advected from the seeds. That

means, given a seed point, a circle with the smallest radius was first shown to the subject, who would then mark the streamline intersection point on the circle. After that, another circle with a larger radius was shown around the same point. This process repeated six times for each seed point. For some seed points, if the subjects believed the advection would go out of boundary or terminate at some point before it reached the circle, such as stagnant points, they could identify the last point in the circle instead of on the circle. Fig. 9 shows a screen snapshot of the interface for this task with only one circle drawn.

Our user study was not timed, so subjects had enough time to give the answers. In summary, the questions involved in our study were: (1) Rate images based on the easiness to follow the underlying flow paths. (2) Rate images based on the easiness to locate the critical points by observing the streamlines. (3) Rate images based on the the overall effectiveness of visualization considering both the flow paths and critical points. (4) Predict where a particle randomly picked up in the field will go in the subsequent steps.

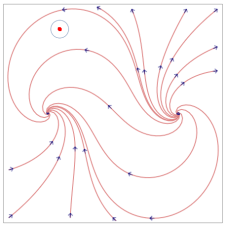


Figure 9: Interface for predicting particle advection paths. Blue arrows on red streamlines show the flow directions. The red point is the particle to be advected from.

5.2.3 Results and Discussions

For the task about rating how easily the streamline images allow the subjects to follow the flow paths, the study result is shown in Table 1. From the result, we can see that most of the subjects prefer images generated by our algorithm. When we analyzed the results from individual subjects in detail, we found that, for some images generated by our algorithm, if they are too abstract, some subjects tended to rate the evenly-spaced based methods higher. Even though the subjects could tell and follow the flow directions with images from our algorithm, evenly spaced methods were better for them to pinpoint the vectors at local points, because the streamlines were uniformly placed and cover all the domain. We also found that six subjects liked our images very much and always rated the highest, while one subject completely did not like all images generated by our algorithm and rated all our images the lowest.

Algorithm	Rank 1	Rank 2	Rank 3
Mebarki et al. 's	5.4%	45.5%	51.0%
Liu et al. 's	20.1%	46.9%	30.0%
Ours	74.5%	7.6%	19.0%

Table 1: The percentages of user rankings for each image based on the easiness to follow the underlying flow paths.

Even though our algorithm does not explicitly place more streamlines near critical points, it indeed captures most of the features around the critical points. This is because vectors around critical points are less coherent and our algorithm is designed to place streamlines based on the streamline coherence. Additionally, streamlines getting converged or diverged around critical points contribute more ink in the neighborhood of them, which makes the critical points much more noticeable. The second question in our first task was to ask the subjects to rank how helpful the streamlines in the images were for the subjects to detect critical points. The study result, shown in Table 2, suggests that images generated from our algorithm are more helpful for the subjects to detect the critical points. This result is in accordance with our expectation since our algorithm allows the viewer to focus on more prominent flow features. Our algorithm allows the streamlines to advect as far as possible once they start. Around critical points, relatively speaking, the streamlines become dense and converge around a small region near each critical point. According to Tufte [12], more data ink should be accumulated around the more important regions.

Algorithm	Rank 1	Rank 2	Rank 3
Mebarki et al. 's	3.3%	42.5%	60.0%
Liu et al. 's	7.7%	52.7%	37.8%
Ours	89%	4.8%	2.2%

Table 2: The percentages of user rankings for each image based on the easiness to locate the critical points by observing the streamlines.

The third question asked the users to rate the overall effectiveness of visualization considering both the flow paths and directions and it let the subjects to decide what they think are more important to visualize a vector field and how to balance the possible conflict between those two criteria. It is possible some images are good at depicting flow paths, while others are good at depicting critical points. The study result is shown in Table 3.

Algorithm	Rank 1	Rank 2	Rank 3
Mebarki et al. 's	3.5%	42.5%	57.0%
Liu et al. 's	19.9%	52.7%	37.8%
Ours	76.6%	4.8%	5.2%

Table 3: The percentages of user rankings for each image based on the overall effectiveness of visualization considering the flow paths and critical points.

For the task about predicting the advection paths of particles, error was measured as the Euclidean distance between the user-selected point and the correct point from the integration using the actual vector data, in the unit of cells. Mean errors are shown in Fig. 10 with error bars depicting plus and minus of the standard deviation. We observe that as the radius of circle was increased, the error became slightly larger. In other words, the closer to the starting seed points, the easier for the subjects to pinpoint the particle path, except when the flow becomes convergent in some regions. In this case, even if the radius of the circle becomes larger, because the space between streamlines becomes smaller, it is still easier for the subjects to locate the advection path. Overall, from the test result we can see that the errors were well bounded. In other words, the subjects were able to predict the flow paths reasonably well given the illustrative streamlines drawn by our algorithm. In general, the error range is related to and constrained by the spacing between streamlines, which depends on how similar the nearby streamlines are.

6 Performance

We have tested our algorithm on a PC with an Intel Core 2 2.66GHz processor, 2 GB memory, and an nVIDIA Geforce 7950 GX2 graphics card with 512 MB of video memory. The streamlines were numerically integrated using a constant step size Runge-Kutta fourth order integrator. In an earlier section, we have presented three comparative results generated by Mebarki et al. 's, Liu et al. 's, and our algorithm in Fig. 8. Generally speaking, algorithms generating evenly-spaced streamlines are fast, and the performance is relatively independent of the flow feature. Our algorithm generates

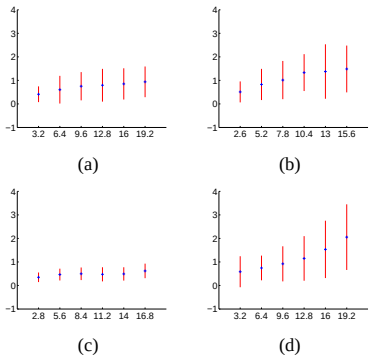


Figure 10: Mean errors for the advection task on the four different datasets. X axis stands for radius of circles around the selected points, and Y axis depicts the mean error plus or minus the standard deviation. Larger value along Y axis means higher error. Y axis starts from -1 to make the graphs easier to visualize. Dimensions of the datasets (a) 64x64 (b) 64x64 (c) 64x64 (d) 100x100.

streamlines by evaluating flow features locally and globally, however, from the timings listed in Table 5 for the four datasets (Table 4), it can be seen that our algorithm can also run at interactive speeds.

There are three main steps in our algorithms: updating distance fields (section 3.1), computing local dissimilarity (section 3.2), and selecting seeds (section 3.5) including computing the global dissimilarity values. Updating distance fields takes place whenever a new streamline is generated. We implemented this on GPUs: for each line segment of the newly generated streamline, a quadrilateral is drawn to a window with the same size as the flow field. The fragment shader computes the distance from each fragment to the line segment. This distance is set to be the depth of the fragment. After all line segments from a streamline are drawn, the depth test supported by the graphics hardware returns the smallest distance from every pixel to the streamline in the depth buffer, which is then read back to the main memory. On the CPU, the distances to the nearest M streamlines for each pixel are recorded. The computation of local dissimilarity is also performed on the CPU by blending the influence of multiple nearby streamlines. From the timings, we can see that when the size of the flow field increases, more time is spent on the portion of our algorithm that runs on the CPU. Although we have not done so, the computation of the dissimilarity metric for each pixel potentially can be implemented on GPUs as well and will be our future work. This could also reduce the overhead of transferring data from CPU to GPU, and reading back from GPU to CPU.

Dataset	Dimension	# of lines	# of line segments
Fig. 8(c)	64x64	18	696
Fig. 6(a)	100x100	19	1204
Fig. 5	400x401	28	3697
Fig. 4	576x291	45	6129

Table 4: Information of four different datasets, and the number of streamlines generated by our algorithm.

7 Conclusions

In this paper, we have presented a seeding strategy to generate streamlines in an illustrative and representative manner for 2D flow fields. Our algorithm fully utilizes the spatial coherence in the underlying flow fields, such that the density of streamlines in the final images can be varied to reflect the coherence of the underlying flow patterns and provide visual focuses. Our method is based on the measurement of dissimilarity between streamlines locally and globally. Our approach is innovative in three regards: (1) the density of streamlines is closely related to the intrinsic flow features of the vector fields, (2) our method does not explicitly rely on detecting the existence of critical points, and (3) the abstract and illustrative visualization generated by our algorithm can effectively reduce visual cluttering. User studies were conducted to evaluate the effectiveness of using illustrative streamlines generated by our algorithm to depict the flow information. Results suggest that users can interpret the flow directions and capture important flow features. In the future, we plan to explore various strategies to measure the similarity between streamlines and other flow features, to further speed up our algorithm using GPUs, and to extend our algorithm for visualizing time-varying flow fields.

Total Timing	Updating Distance Field	Computing Local dissimilarity	Finding Seeds
0.078	0.031	0.00	0.047
0.156	0.079	0.031	0.046
2.562	0.799	1.355	0.172
4.453	1.08	1.639	1.375

Table 5: Timings (in seconds) measured for generating streamlines using our algorithm. Each row corresponds to a data set listed in the same row of Table 4.

References

[1] U. Bordoloi and H. Shen. Hardware accelerated interactive vector field visualization: A level of detail approach. *Computer Graphics Forum (Proceedings of Eurographics 2002)*, 21(3):605–614, 2002.

[2] B. Cabral and C. Leedom. Imaging vector fields using line integral convolution. In *SIGGRAPH '93*, pages 263–270, 1993.

[3] Q. Du and X. Wang. Centroidal voronoi tessellation based algorithms for vector fields visualization and segmentation. In *IEEE Visualization*, pages 43–50, 2004.

[4] S. F. F. Gibson. Using distance maps for accurate surface representation in sampled volumes. In *VVS '98*, pages 23–30, 1998.

[5] B. Heckel, G. Weber, B. Hamann, and K. I. Joy. Construction of vector field hierarchies. In *IEEE Visualization*, pages 19–26, 1999.

[6] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. In *Visualization in Scientific Computing*, pages 43–56, 1997.

[7] M. Jones, J. Baerentzen, and M. Sramek. 3d distance fields: A survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):581–599, 2006.

[8] D. Laidlaw, R. Kirby, C. Jackson, J. Davidson, T. Miller, M. Silva, W. Warren, and M. Tarr. Comparing 2d vector field visualization methods: A user study. *IEEE Transactions on Visualization and Computer Graphics*, 11(1):59–70, 2005.

[9] Z. Liu, R. Moorhead, and J. Groner. An advanced evenly-spaced streamline placement algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):965–972, 2006.

[10] A. Mebarki, P. Alliez, and O. Devillers. Farthest point seeding for efficient placement of streamlines. In *IEEE Visualization*, pages 479–486, 2005.

[11] A. Telea and J. Wijk. Simplified representation of vector fields. In *IEEE Visualization*, pages 35–42, 1999.

[12] E. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1986.

[13] E. Tufte. *Envisioning Information*. Graphics Press, 1990.

[14] G. Turk and D. Banks. Image-guided streamline placement. In *Proceedings of SIGGRAPH '96*, pages 453–460, 1996.

[15] J. J. van Wijk. Spot noise texture synthesis for data visualization. In *SIGGRAPH '91*, pages 309–318, 1991.

[16] J. J. van Wijk. Image based flow visualization. In *SIGGRAPH '02*, pages 745–754, 2002.

[17] V. Verma, D. Kao, and A. Pang. A flow-guided streamline seeding strategy. In *IEEE Visualization*, pages 163–170, 2000.