An Advanced Evenly-Spaced Streamline Placement Algorithm

Zhanping Liu, Robert J. Moorhead II, *Senior Member*, *IEEE*, and Joe Groner, *Student Member*, *IEEE*

Abstract—This paper presents an advanced evenly-spaced streamline placement algorithm for fast, high-quality, and robust layout of flow lines. A fourth-order Runge-Kutta integrator with adaptive step size and error control is employed for rapid accurate streamline advection. Cubic Hermite polynomial interpolation with large sample-spacing is adopted to create fewer evenly-spaced samples along each streamline to reduce the amount of distance checking. We propose two methods to enhance placement quality. Double queues are used to prioritize topological seeding and to favor long streamlines to minimize discontinuities. Adaptive distance control based on the local flow variance is explored to reduce cavities. Furthermore, we propose a universal, effective, fast, and robust loop detection strategy to address closed and spiraling streamlines. Our algorithm is an order-of-magnitude faster than Jobard and Lefer's algorithm [8] with better placement quality and over 5 times faster than Mebarki et al.'s algorithm [9] with comparable placement quality, but with a more robust solution to loop detection.

Index Terms—Flow visualization, evenly-spaced streamlines, streamline placement, seeding strategy, closed streamlines.

1 Introduction

There has been significant research on texture-based flow visualization such as spot noise [1], LIC [2], Fast LIC [3], UFLIC [4, 5], LEA [6], and IBFV [7], to name a few. However, obtaining visually pleasing images requires an intrinsically huge computational expense for pixel / texel based synthesis, which makes it difficult for texture-based techniques to meet the increasing resolution of real-world applications. Streamlines remain one of the most straightforward and fastest visualization techniques used to investigate flow phenomena. One major problem is that either a local discrete coarse view or a cluttered image of the flow is obtained unless an appropriate seeding strategy is employed. The key goal of flow visualization is to provide insight into flow structures while an aesthetic image filled with evenly-spaced streamlines helps the user quickly recognize flow patterns without visual distractions that would arise from arbitrarily-advected lines. Furthermore, a layout with longer streamlines is a more effective representation than one with shorter streamlets since discontinuities in the latter impair the impression of a flow field. Due to flow divergence and convergence, there is a tradeoff between placement of evenly-spaced streamlines and advection of long streamlines, with grid-point based arrow plots or short streamlets at one extreme and un-controlled streamlines at the other extreme. Thus there is no single global solution to the placement of strictly evenly-spaced streamlines for a flow with divergence or convergence. One important objective is to strike a balance between the two factors to provide an optimal distribution of evenly-spaced streamlines.

In this paper we present a novel streamline placement algorithm. A fourth-order Runge-Kutta integrator allows rapid, but accurate, flow line advection. Cubic Hermite polynomial interpolation allows fewer evenly-spaced samples along each streamline. Double queues are used to prioritize topological seeding and to favor long streamlines to minimize discontinuities. Adaptive distance control driven by the local flow variance is exploited to reduce cavities. In addition, we present a fast robust loop detection strategy to address closed and spiraling streamlines for complex flow fields. Interactive placement of high-density streamlines can be achieved for large flow fields on an ordinary PC.

The remainder of this paper is organized as follows. Section 2 gives an introduction to existing algorithms. Section 3 presents our advanced evenly-spaced streamline placement algorithm. Results are given in section 4 to demonstrate the placement speed, quality, and robustness of our algorithm. We conclude the paper with a brief summary and outlook on future work.

2 Previous Work

Turk and Banks [10] presented an image-guided algorithm that iteratively refines an initial placement by creating, merging, repositioning, lengthening, or shortening streamlines to minimize an energy function. The energy function describes the difference between the current placement and a desired one. This algorithm generates high-quality placements but the huge computational cost restricts its practical applicability. Later Mao et al. [11] extended it for streamline placement on curved surfaces.

Other algorithms use inter-sample distance control to approximate inter-line distance control in streamline advection. They mainly differ in the greedy seeding strategy that drives the placement process until a queue of candidate seeds is empty. Jobard and Lefer [8] proposed to incorporate an FIFO neighborhood seeding strategy with a cell-based distance controller for evenly-spaced streamline placement (Fig. 1). Their algorithm can create a variety of streamline placements, ranging from dense texture-like to sparse hand-drawing styles, by simply adjusting the separating distance between adjacent streamlines. The distance controller checks each streamline sample (including seeds) using a Cartesian grid, which, with the cell size equal to the separating distance, is superimposed on the flow field. A pointer is added to the sample-pointer list of the containing cell if the sample is greater than the threshold distance to the samples already accepted in the nine local cells. As samples are successively generated and evenly spaced along each streamline through a fixed step size integrator, each of them is checked by the distance controller and any sample-refusal terminates streamline advection. Each streamline, once advected long enough to survive, is saved and introduces two neighboring candidate seeds in the local orthogonal flow direction to a global queue, from which a candidate is extracted to begin a new advection unless rejected by the controller. This neighborhood seeding strategy produces artifacts in the placement. Noticeable discontinuities tend to occur in turbulent and even laminar areas due to the rigid seeding scheme and the FIFO

• *Zhanping Liu, Robert J. Moorhead II, and Joe Groner are with HPC² / GRI / Visualization Analysis and Imaging Lab, PO Box 9627, Mississippi State University, MS 39762-9627. E-Mail: zhanping, rjm, bjg@hpc.msstate.edu.*

Manuscript received 31 March 2006; accepted 1 August 2006; posted online 6 November 2006.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

scheduler. Large cavities may remain in the placement, particularly around critical points, due to the consecutive terminations of sequentially advected streamlines. This algorithm was later adapted for unsteady flow fields [12], extended to multi-resolution placement [13], and combined with streamline illumination for 3D flows [14].

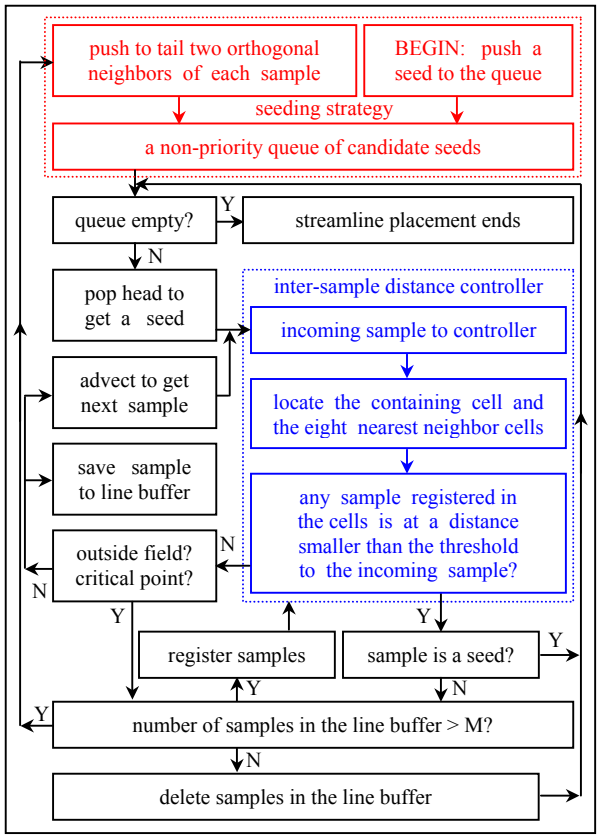

Fig. 1. The pipeline of a typical streamline placement algorithm based on inter-sample distance control. An FIFO neighborhood seeding strategy is incorporated with a cell-based distance controller.

Verma et al. [15] presented a flow-guided seeding strategy by which topological templates are designed to choose seeds around critical points before Poisson-disk distribution is used to place additional seeds elsewhere. Topological features of the flow can be retained in a placement even when the global density is low. However, the global uniformity is degraded and the density disparity gets exacerbated as the separating distance increases. This work was later extended to 3D streamline placement [16]. Recently Mebarki et al. [9] proposed a farthest point seeding strategy that chooses the circumcircle center of the largest triangle as the new seed point from a priority queue of unvisited Delaunay triangles. For each advected streamline, the unvisited incident triangles with circumcircle diameters larger than the separating distance are pushed to the priority queue and sorted by the diameter. This algorithm and [8, 15] are all based on inter-sample distance control and the major difference lies in the greedy seeding strategy. These algorithms run at approximately the same speed, with a minor variation caused by the specific seeding strategy, implementation, and measurement platform. They all adopt a fixed step size integrator to create evenly-spaced samples for distance control. As a consequence, they cannot achieve interactive placement of high density due to the large cost of both streamline advection and distance checking on a significant number of samples, nor can they robustly detect looping streamlines due to insufficient numerical accuracy. They are restricted to small-size, non-turbulent flow fields (e.g., very small-size synthetic data) for which numerical inaccuracy and the large computational cost are made less noticeable. They are not well suited for visualizing large, complex, real-world flow fields.

3 Advanced Evenly-Spaced Streamline Placement

In this section, we present an *Advanced Evenly-Spaced Streamline* (ADVESS) placement algorithm for fast, high-quality, and robust layout of flow lines. First we accelerate placement generation through rapid streamline advection and significant reduction of the cost of cell-based inter-sample distance control. The former is achieved by using a fourth-order Runge-Kutta integrator with adaptive step size and error control, or RK4-ASSEC for short. The latter is accomplished by using cubic Hermite polynomial interpolation with large sample-spacing to create fewer evenly-spaced samples along each streamline. We also enhance placement quality via double queues and adaptive distance control. Double-queuing prioritizes seeds around critical points and seeds introduced by long streamlines to combat discontinuities. Adaptive distance control based on the local flow variance places streamlines where cavities would otherwise remain. Finally, we present a universal, effective, fast, and robust loop detection strategy.

3.1 Accelerating Placement Generation

The inter-sample distance control scheme mandates that the interval between any two successive samples of a streamline be smaller than the specified separating distance to make this proxy distance control acceptable. Any fixed step size integrator is a tradeoff between computational speed and numerical accuracy. We thus adopt an RK4-ASSEC integrator and cubic Hermite polynomial interpolation with large sample-spacing to accelerate placement generation while maintaining high numerical accuracy. Other schemes [20] may be considered to achieve the same goal.

3.1.1 Accelerating Streamline Integration

A streamline is a curve everywhere tangent to the flow, i.e.,

$$d\rho(\tau) / d\tau = \upsilon(\rho(\tau))$$

where $\rho(\tau)$ is a point along the streamline and $\upsilon(\rho(\tau))$ is the flow vector at that point. Given a seed with $\tau = 0$, the streamline is obtained by solving the above differential equation step by step:

$$\rho(\tau + \Delta\tau) = \rho(\tau) + \int_{\tau}^{\tau+\Delta\tau} \upsilon(\rho(\tau))d\tau$$

When vectors obtained through interpolation are always normalized, as is the case with our ADVESS algorithm, the integration variable τ is equal to and hence can be used as the curve length. The steps of RK4, a well-known numerical integrator, are:

$$\Delta\rho_0 = s \times \upsilon(\rho(\tau))$$

$$\Delta\rho_1 = s \times \upsilon(\rho(\tau) + \Delta\rho_0 / 2)$$

$$\Delta\rho_2 = s \times \upsilon(\rho(\tau) + \Delta\rho_1 / 2)$$

$$\Delta\rho_3 = s \times \upsilon(\rho(\tau) + \Delta\rho_2)$$

$$\rho(\tau + s) = \rho(\tau) + \Delta\rho_0 / 6 + \Delta\rho_1 / 3 + \Delta\rho_2 / 3 + \Delta\rho_3 / 6$$

where $s = \Delta\tau$ is the integration step size.

Adaptive step size based on local error control [3] can be added to the RK4 integrator to perform fast, but accurate, integration. An efficient error estimator based on embedded RK formulae is:

$$\varepsilon = \Delta\rho_3 / 6 - \upsilon(\rho(\tau + s)) \times s / 6$$

Given an error tolerance range $(\varepsilon_{min}, \varepsilon_{max})$, the step size is doubled when $\varepsilon < \varepsilon_{min}$ and halved when $\varepsilon > \varepsilon_{max}$. An RK4-ASSEC integrator as described above can be very fast and highly accurate.

3.1.2 Evenly Sampling Streamlines

Due to adaptive step size, the distance between two successive points obtained through RK4-ASSEC, $\rho(\tau_n)$ and $\rho(\tau_{n+1})$, is not fixed. To

create evenly-spaced samples for distance control, we use cubic Hermite polynomial interpolation to sample each streamline:

$$\Phi(\mu) = A\mu^3 + B\mu^2 + C\mu + D; \quad \mu = (\tau_s - \tau_n) / (\tau_{n+1} - \tau_n) \in [0, 1]$$

where τ_n and τ_{n+1} are the curve lengths (as discussed in section 3.1.1) for points $\rho(\tau_n)$ and $\rho(\tau_{n+1})$, respectively. $\tau_s \in [\tau_n, \tau_{n+1}]$ is the curve length for an evenly-interpolated sample and:

$$A = 2\Phi(0) - 2\Phi(1) + \Phi'(0) + \Phi'(1); \qquad C = \Phi'(0)$$
$$B = -3\Phi(0) + 3\Phi(1) - 2\Phi'(0) - \Phi'(1); \qquad D = \Phi(0)$$

with boundary conditions:

$$\Phi(0) = \rho(\tau_n); \qquad \Phi'(0) = (\tau_{n+1} - \tau_n)\upsilon(\rho(\tau_n))$$
$$\Phi(1) = \rho(\tau_{n+1}); \qquad \Phi'(1) = (\tau_{n+1} - \tau_n)\upsilon(\rho(\tau_{n+1}))$$

τ_s is initialized to a given sampling size ζ and is incremented by ζ following each sample interpolation that occurs when τ_s falls within $[\tau_n, \tau_{n+1}]$. The interval of two successive points, i.e., the step size $\tau_{n+1} - \tau_n$, may be tens of times the field cell size and hence there may be many evenly-spaced samples in the range, which can be quickly generated using forward difference equations:

$$\Phi_1(\mu) = \Phi(\mu + \mu_\delta) - \Phi(\mu) = 3A\mu_\delta\mu^2 + (3A\mu_\delta^2 + 2B\mu_\delta)\mu + A\mu_\delta^3 + B\mu_\delta^2 + C\mu_\delta$$
$$\Phi_2(\mu) = \Phi_1(\mu + \mu_\delta) - \Phi_1(\mu) = 6A\mu_\delta^2\mu + 6A\mu_\delta^3 + 2B\mu_\delta^2$$
$$\Phi_3(\mu) = \Phi_2(\mu + \mu_\delta) - \Phi_2(\mu) = 6A\mu_\delta^3 = \text{constant}$$

where $\mu_\delta = \zeta / (\tau_{n+1} - \tau_n)$. Once $\Phi_1(\mu)$, $\Phi_2(\mu)$, and $\Phi_3(\mu)$ are computed for the first evenly-spaced sample within an interpolation interval, the subsequent samples within the same interval can be recursively obtained using three additions per sample:

$$\Phi(\mu_{k+1}) = \Phi(\mu_k) + \Phi_1(\mu_k)$$
$$\Phi_1(\mu_{k+1}) = \Phi_1(\mu_k) + \Phi_2(\mu_k)$$
$$\Phi_2(\mu_{k+1}) = \Phi_2(\mu_k) + \Phi_3(\mu_k)$$

where $\Phi(\mu_k)$ and $\Phi(\mu_{k+1})$ are evenly-interpolated samples k and k+1 within the same interval, respectively.

3.1.3 Generating Fewer Samples

Two distance parameters are usually used for inter-sample distance control to favor long streamlines. d_{sep} is the separating distance given by the user, which specifies the minimum distance between seeds and streamlines. d_{test}, a percentage of d_{sep}, serves as the minimum distance between regular (non-seed) samples and streamlines. $d_{test} = 0.5d_{sep}$ usually gives good visual results, but this depends on the characteristics of the flow being visualized. A single distance parameter would create a distracting placement with lots of short streamlets due to the early termination of advection by excessively stringent distance control on regular samples. The consequence of using two distance parameters is a disparity in streamline density. As mentioned in section 1, there is a tradeoff between placement uniformity and long streamline generation (or placement continuity) that cannot be entirely solved. This involves *Proximity Tolerance Degree* (PTD), a metric used in our analysis of density disparity to describe the extent to which the actual inter-line distance may be smaller than the specified separating distance. When $d_{test} = 0.5d_{sep}$, the PTD is larger than $d_{sep} / d_{test} = 2$. The PTD is usually larger than 1 and even much larger to maintain the placement continuity. The PTD results from an implicit influence by the approximation of inter-sample distance control to inter-line distance control, in combination with an explicit influence by the percentage of d_{test} to d_{sep}.

Our ADVESS algorithm uses cubic Hermite polynomial interpolation to generate evenly-spaced samples along each streamline. We choose the sampling size equal to d_{test}, which greatly reduces the cost of distance checking due to fewer samples generated, particularly when the separating distance increases. Such a large sampling size used in ADVESS (Fig. 2a) tends to result in a larger PTD than does a smaller fixed step size used in other algorithms (Fig. 2b) because the former provides less accurate distance control than the latter. Roughly speaking, the PTD caused by the former is λ ($1 < \lambda < 1.155 = 1 / \cos 60°$) times the one by the latter and the worst case occurs when $SA_1 = SB_1 = A_1B_1 = d_{test}$. In order to maintain acceptable global uniformity, a larger d_{test} can be used in ADVESS to suppress the PTD.

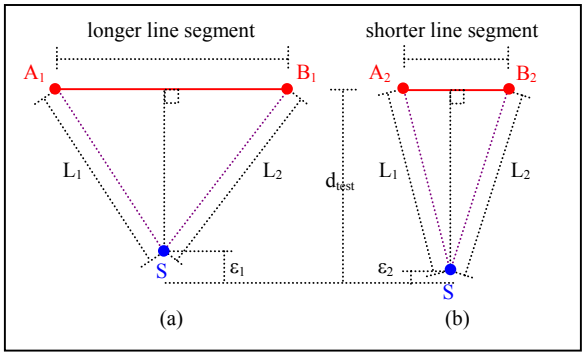

Fig. 2. Given a sample S and the distances to two samples (A_1, B_1 in Figure a and A_2, B_2 in Figure b) of a streamline, i.e., L_1 and L_2 ($L_1 >= d_{test}$, $L_2 >= d_{test}$), a longer inter-sample interval (Figure a, the case for ADVESS) introduces a larger distance control error ($\varepsilon_1 > \varepsilon_2$) than does a shorter inter-sample interval (Figure b, the case for other algorithms using a fixed step size integrator).

Since ADVESS employs a large sampling stride, there may be a large curvature between two successive evenly-spaced samples as opposed to a straight line segment shown in Fig. 2a. To address this scenario for acceptable distance control, distance checking is performed not only on evenly-spaced samples obtained through cubic Hermite polynomial interpolation, but also on raw points generated by the RK4-ASSEC integrator. Once a streamline is advected and survives the length check, all these points are registered in the distance controller while only raw points are rendered to show the placement. Even with raw points considered, many fewer samples are created in ADVESS than in other algorithms and the computational cost of distance checking is considerably reduced.

3.2 Enhancing Placement Quality

Since the adaptive step size may be large when streamlines cross the field boundaries, streamlines must be clipped against the boundaries. Otherwise the placement would be cluttered by simply truncated streamlines and accordingly encroaching streamlets (upper left in Fig. 3). Evenly-spaced sampling, distance checking, and controller registration must be performed on the extended part of each clipped streamline. Otherwise the encroaching streamlets would remain due to being accepted by the un-updated controller (upper right in Fig. 3). As boundary artifacts are eliminated via streamline clipping coupled with normal distance control, there are still discontinuities and cavities (lower left in Fig. 3) in the placement. To enhance placement quality, we present double-queuing and adaptive distance control to minimize discontinuities and cavities, respectively.

3.2.1 Double-Queuing Seeds

The random selection of initial seeds and the FIFO seed scheduler adopted in Jobard and Lefer's algorithm [8] neglect the importance of flow topology and the desire for long streamlines. Verma et al.'s seeding strategy [15] emphasizes topological structures but sacrifices the global uniformity. Mebarki et al.'s seeding strategy [9] based on Delaunay triangulation is too computationally expensive to handle a huge number of samples when a small fixed step size is used for high-density streamline placement.

We propose an improved seeding strategy that prioritizes topological seeding and favors long streamlines. A lightweight topological analysis is conducted to provide a guide to select initial seeds based on topological templates [15]. To spread this schematic

seeding pattern as much as possible, we assign a higher priority to streamline seeds than regular (non-seed) samples in introducing neighboring candidates. Specifically, we employ two queues, a primary queue and a secondary queue. The primary queue serves as the major seed provider where initial topological seeds are assigned a highest weight. Candidates introduced by streamline seeds are also stored in the primary queue, but are sorted by the weight, i.e., the streamline length. The secondary queue stores candidates introduced by regular samples in an FIFO manner and it is used only when the primary queue is temporarily empty. Our ADVESS algorithm runs until the secondary queue is empty. The pseudo code for double-queuing seeds is given in Fig. 4.

Double-queuing enhances placement quality by reducing discontinuities and favoring long streamlines. The placement generated using double queues (lower right in Fig. 3) is better in quality than the one generated using a simplistic neighborhood seeding strategy (lower left in Fig. 3). Double-queuing adds a marginal cost to ADVESS since a lightweight topological analysis is conducted and sorting is performed only in the primary queue on a small number of candidates introduced by streamline seeds.

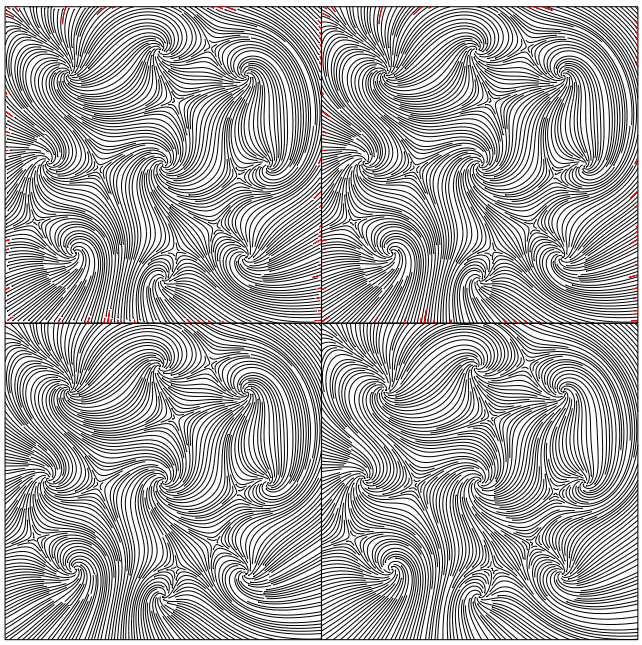

Fig. 3. Placements of 1% density generated for a 300 × 300 flow field. Streamlines are simply truncated outward and accordingly streamlets (in red) encroach inward near the boundaries when streamlines are not clipped against the boundaries (upper left). When streamlines are clipped without distance control and controller registration, the encroaching streamlets (in red) remain due to being accepted by the un-updated controller (upper right). When streamlines are clipped with normal distance control, boundary artifacts are eliminated. There are still a lot of discontinuities and cavities in the placement generated using a simplistic neighborhood seeding strategy (lower left). Discontinuities are significantly reduced and streamlines get longer in the placement generated using double queues, though there are still some cavities, particularly near critical points (lower right).

3.2.2 Adaptive Distance Control

Double-queuing alleviates the cavity problem, though there are still cavities that cannot be eliminated (lower right in Fig. 3). Thus we propose to use adaptive distance control to attack this problem. This idea was inspired by the use of two distance parameters, i.e., d_{sep} and d_{test}, in many streamline placement algorithms based on inter-sample distance control. The second parameter relaxes distance control, helping streamlines pass where cavities might otherwise remain. Fixing the value of d_{test} throughout the streamline placement process fails to consider the flow behavior. As a result, inter-sample distance control is still excessively stringent in some turbulent areas while it is unnecessarily loose in some laminar areas. The disparity of the placement density is largely caused by such a stiff distance control scheme.

```
void PopToAdvect(QUEUE* q)
{ Pop a seed s from q head for streamline advection;
  if(s is accepted && (l = streamline length) > L)
  { s introduces two neighboring candidates c1 and c2 with weight l;
    Send c1 and c2 to primary queue and sort them by weight;
    for(each regular non-seed sample of the streamline)
      Push two neighboring candidates to secondary queue tail;
  }
}
void DoubleQueuingSeeds()
{ Extract critical points and use seeding templates to set initial seeds;
  Add initial seeds to primary queue with a maximum weight MAX;
  do
  { while(primary queue not empty) PopToAdvect(primary queue);
    if(secondary queue not empty) PopToAdvect(secondary queue);
  } while(secondary queue not empty);
}
```

Fig. 4. Pseudo code for double-queuing seeds.

Our ADVESS algorithm employs adaptive distance control based on the local flow variance. The local variance of a two-dimensional flow can be measured by a scalar value:

$$\text{conDiv} = \partial(v_x) / \partial x + \partial(v_y) / \partial y$$

where v_x and v_y are vector components. The value is positive when the flow diverges and negative when the flow converges. Since streamlines are advected in both directions, the magnitude of the local flow variance, $|\text{conDiv}|$, may be used to govern adaptive distance control. It is computed at each grid point and mapped, usually non-linearly, to [0, 1] before it is used to determine an *adaptive map*. The adaptive map stores scaling parameters, which are used to modulate the separating distance:

$$\text{adaptiveMap}[i] = \textit{minScale} + (1 - \textit{minScale})(1 - \text{varianceMag}[i])$$

where *minScale* is a minimum scaling value specified by the user. The adaptive map value ([*minScale*, 1]) is larger for laminar areas and smaller for turbulent areas. During the placement process, the adaptive map is accessed to adjust distance control.

Care should be taken not to make *minScale* so small that it affects the global uniformity. Currently we use the adaptive map to modulate both d_{sep} and d_{test}. For the 300 × 300 flow field used in Fig. 3, we set *minScale* to 0.75 with good results. The placement generated using adaptive distance control is shown in Fig. 8 (bottom middle). Cavities are effectively eliminated and placement quality is considerably improved with acceptable global uniformity at a negligible additional cost.

3.3 Detecting Streamline Loops

Without effective loop detection, a placement might be cluttered by closed or spiraling (not closed) streamlines (upper in Fig. 5). Algorithms based on inter-sample distance control are susceptible to this problem if each sample is checked against only the samples of other streamlines. To perform loop detection, the sample cannot be simply checked against other samples of the current streamline. Otherwise the advection would be immediately terminated even when there is no loop because the distance between any two successive samples is not greater than the separating distance.

Although there are general algorithms [17, 18] for very accurate detection of closed streamlines, they are too computationally expensive to be adapted for interactive streamline placement. Mebarki et al. [9] proposed to insert all samples in a Delaunay triangulation by which the minimum circumcircle diameter of the

triangles caused by the new sample is used to approximate both inter-streamline distance and streamlineself distance (for loop detection). This strategy mandates that the step size be significantly smaller than the separating distance for acceptable approximation, which causes a huge computational cost. Given an unknown flow field, this method is also prone to missing loops when the size is not sufficiently small. A demo program [19] of Mebarki et al.'s algorithm fails to handle the flow field used in Fig. 5.

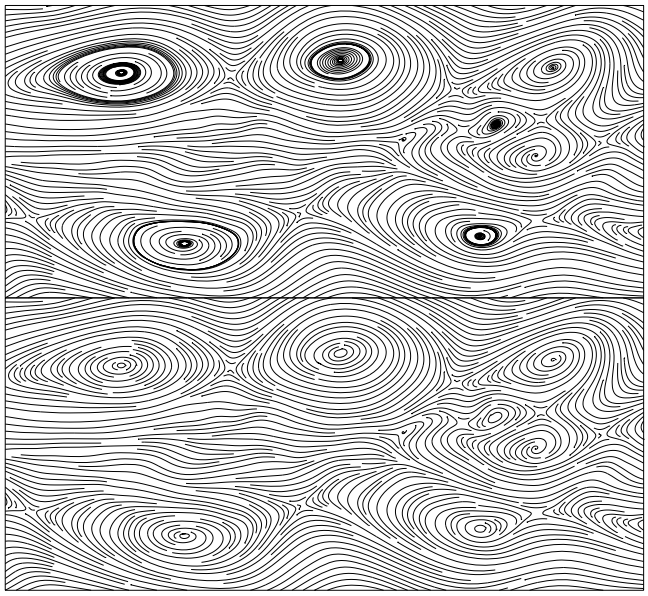

Fig. 5. A placement generated without loop detection is cluttered by streamline loops (upper). Closed and spiraling streamlines can be robustly detected and effectively processed using our ADVESS (lower). Flow field size: 525 × 241; separating distance: 4.5 in cell size.

We propose a fast effective loop detection strategy, which is robust due to high accuracy guaranteed through an RK4-ASSEC integrator. A streamline is *ill-looping* when the distance between two successive cycles is smaller than the threshold distance d_{test}. A closed streamline is always ill-looping as a whole. However, a spiraling streamline may be ill-looping only in part of the curve. Our principle is to strictly disallow *the ill-looping part* of a streamline, if any, to advect more than one cycle due to the cluttering it would cause otherwise. Within this one-cycle limit, the ill-looping part is allowed to advect as long as possible to encourage the formation of closed loops, without cluttering the placement. It is worth mentioning that regardless of the metrics used, effective loop detection cannot be achieved by simply checking the current sample against the seed since the ill-looping part of a spiraling streamline does not necessarily begin with the seed. To address this problem, a brute-force but universal solution is to check each existing sample against the current sample with respect to the distance and *the ill-looping angle*. The ill-looping angle of each existing sample relative to the current sample is not obtained through segment-wise angle accumulation due to the large computational cost. Instead it is indirectly measured by a dot-product ($\odot$) between two normalized vectors. In fact, the major task of loop detection is to distinguish between an *opponent sample* p_n (Fig. 6) and a *proponent sample* p_m when both are less than d_{test} to the current sample p_0 *and* the two associated flow vectors, v_n and v_m, are both aligned, within a small threshold angle α, with the latest *uni-directional* (i.e., always in the positive flow direction) line segment vector v_1. p_n, as a "long-term α-neighbor" of p_0, approaches the one-cycle limit while p_m, as a "short-term α-neighbor" of p_0, does not. The two vectors, pointing to p_0 and the previous sample p_1, respectively, from any opponent or proponent sample, are used to compute two dot-products with v_1. p_n is then distinguishable from p_m by at least one negative dot-product for positive advection. This holds for negative advection if the two dot-products are negated. There may be several proponent samples found through a loop detection process (one process for each newly generated sample). However, any opponent sample terminates such a process as the ill-looping part has been advected enough (ill-looping angle >= 2π-α) to classify the loop by testing whether the vector pointing from p_0 to p_n is aligned, within a small threshold angle β, with the actual integration vector at p_0. Satisfactory results can be obtained by using $\alpha = 20^\circ$ and $\beta = 10^\circ$ in our test. Despite this loop classification scheme being less accurate than [17, 18], our loop detection strategy is robust enough to avoid any cluttering. Fig. 6 shows the two possible cases for outward negative advection.

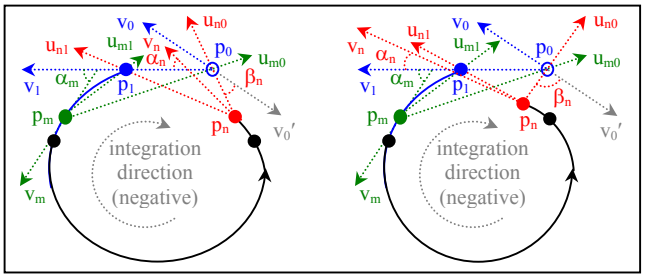

Fig. 6. Given the current sample p_0 and the previous sample p_1 of an outward-negatively advected streamline, the major task of loop detection is to distinguish between an opponent sample p_n ($|p_0p_n| < d_{test}$ and $\alpha_n <= \alpha$; any such sample terminates the loop detection process) and a proponent sample p_m ($|p_0p_m| < d_{test}$ and $\alpha_m <= \alpha$; there may be several samples of this kind found through a loop detection process). p_n is distinguishable from p_m since at least one of the two negated (due to the negative advection) dot-products is negative ($-(u_{n0} \odot v_1) < 0$, $-(u_{n1} \odot v_1) < 0$ on the left and $-(u_{n0} \odot v_1) > 0$, $-(u_{n1} \odot v_1) < 0$ on the right) for p_n while both are non-negative ($-(u_{m0} \odot v_1) > 0$, $-(u_{m1} \odot v_1) > 0$) for p_m. Samples along the black part of the streamline are not "α-neighbors" of p_0 and hence are skipped. Segment p_0p_1 and α are exaggerated here for illustrative purposes.

We use a loop controller, with a data structure similar to that of the global controller (Fig. 1), to achieve fast loop detection by limiting distance checking against the current sample to only the samples in the nine local cells. Furthermore, the signed RK4-ASSEC step index, determined by the advection direction, is attached to each raw sample as the stamp. Each evenly-interpolated sample is attached with a neighboring raw sample's stamp. Given a threshold stamp-difference δ, many proponent samples are skipped through fast stamp checking without distance checking against the current sample. Each cell of the loop controller maintains the minimum and maximum stamps of the accepted samples and hence many cells can be efficiently skipped. A conservative value for δ can be 4 and a larger one may be used due to high curve-smoothness guaranteed by adaptive step size. The current sample is first checked by the loop controller and, unless an ill-looping cycle is found, sent to the global controller for inter-streamline distance checking. Otherwise, the advection is terminated to either form a closed loop or refuse further spiraling. The loop controller is cleared upon the advection of each new streamline. Fig. 7 shows the pseudo code for loop detection.

Our loop detection strategy is universal, effective, fast, and robust. Function pointers are used in our implementation to replace all if-statements for high-performance execution. It takes 0.253 seconds for ADVESS with loop detection to generate the placement in Fig. 5 (lower). All loops are detected and processed well.

4 Results and Discussions

We have implemented Jobard and Lefer's algorithm [8] and our ADVESS algorithm using Visual C++ on a Windows XP / Dell Inspiron 2650 notebook (Pentium IV 1.70GHz, 512MB RAM) with exactly the same code on vector interpolation, distance controller, and data structures for storing streamlines. Compared to Jobard and

Lefer's algorithm that uses a fixed step size RK4 integrator and a neighborhood seeding strategy, our ADVESS algorithm employs an RK4-ASSEC integrator coupled with cubic Hermite polynomial interpolation, streamline clipping, double-queuing, adaptive distance control, and loop detection. We have not implemented Mebarki et al.'s algorithm since we have access to the executable program [19]. We have compared the three algorithms on the WinTel platform mentioned above in placement quality and generation speed using a 300 × 300 flow field. The adaptive step size is initialized to 0.0625 and varies between 0.0005 and 32.0 (in cell size) in our algorithm while a fixed step size of 0.15 is set in the other two algorithms. Fig. 8 shows comparative streamline placements for three increasing densities. Placements generated by our algorithm are better than those by Jobard and Lefer's algorithm, and are comparable with those by Mebarki et al.'s algorithm in quality, specifically regarding uniformity, continuity, and coverage. It is worth mentioning that placement quality is heavily dependent on loop detection as demonstrated in Fig. 5, in addition to the above three factors. The flow field used in Fig. 8 is intended to provide a "clean" comparison of streamline placements only in the former three factors, without being cluttered by ill-looping streamlines. Table 1 shows the timings of the algorithms for generating the placements. Such a high placement speed of our algorithm comes along with high accuracy, i.e., 0.0 ~ 0.00001 (in cell size).

```
SAMP { POINT p;      VECTOR v;      short s;    /* s: stamp */ }
CELL { SAMP** sampleList; short s[2]; /* s: min, max stamps */ }

/* p1, p0: previous and current samples;          cp: closing point */
/* a:      negative(–1) / positive(1) advection; cc: containing cell */
/* ⊙:      dot-product;  ◈: normalize vectors and then dot-product */
int LoopDetection(SAMP* p1, SAMP* p0, POINT* cp, int a)
{ for(each of the nine local cells: CELL cell)
  { if(cell.sampleList is empty) continue;
    if(|cell.s[(a+1)/2] – p0→s| < δ) continue;
    for(each sample in cell: SAMP* q = cell.sampleList[i])
    { if(|q→s – p0→s| < δ) continue;
      if(Distance(q→p, p0→p) >= d_test) continue;
      if(Distance(q→p, p0→p) <= ε) { *cp = q→p; return closed; }
      VECTOR v1 = (p0→p–p1→p)*a; if(q→v◈v1 < cosα) continue;
      VECTOR u0 = (p0→p – q→p)*a, u1 = (p1→p – q→p) * a;
      if(u0⊙v1 >= 0 && u1⊙v1 >= 0) continue;
      if(|u0◈p0→v| > cosβ) { *cp = q→p; return closed; }
      return spiraling;
    }
  }
  cc.s[(a+1)/2] = p0→s;   add p0 to cc.sampleList;   return pass;
}
```

Fig. 7. Pseudo code for our loop detection strategy.

Table 1. Timings measured on Pentium IV (1.70GHz, 512MB RAM).

density (% width)	Jobard-Lefer algorithm	Mebarki et al.'s algorithm	our ADVESS algorithm
1.0%	1.882 s	1.262 s	0.241 s
1.5%	1.382 s	0.811 s	0.140 s
2.0%	1.142 s	0.621 s	0.080 s

Fig. 9 shows the streamline placement (upper) generated using our algorithm for a 468 × 337 flow field obtained through a US Navy model of the Northeast Pacific Ocean (180° W ~ 78° W, 20° N ~ 62° N). The land is shown in brown and Hawaii islands in red. The flow is highly turbulent with hundreds of critical points. Our algorithm is robust enough to detect and process all streamline loops. Also shown in Fig. 9 are the two placements generated using our algorithm (lower left) and Mebarki et al.'s algorithm (lower right) for a 201 × 201 flow field that is obtained by zooming a region of interest (184 ~ 284, 136 ~ 236, marked by the red square) of the Northeast Pacific Ocean. There are three spiraling streamlines (marked by blue rectangles) detected by both algorithms while there are two loops (marked by red rectangles) missed by Mebarki et al.'s algorithm that clutter the placement. These two loops can be effectively detected by our algorithm and processed as a closed streamline and a spiraling streamline, respectively. For this zoomed flow, our algorithm is more robust in loop detection and 4.6 times faster in placement generation than Mebarki et al.'s algorithm.

5 Conclusions and Future Work

We have presented an advanced evenly-spaced streamline placement algorithm. It exploits a fourth-order Runge-Kutta integrator with adaptive step size and error control for rapid, but accurate, flow advection. Cubic Hermite polynomial interpolation with large sample-spacing is employed to reduce the amount of distance checking. Double queues are used to prioritize topological seeding and to favor long streamlines to minimize discontinuities. Adaptive distance control based on the local flow characteristics is adopted to reduce cavities. Furthermore, we have presented a universal, effective, fast, and robust loop detection strategy to address closed and spiraling streamlines. In particular, our algorithm is an order-of-magnitude faster than Jobard and Lefer's algorithm [8] with better placement quality. Our algorithm is over 5 times faster than Mebarki et al.'s algorithm [9] with comparable placement quality, but with a more robust solution to loop detection.

As for future work, we would like to extend our ADVESS algorithm to view-dependent streamline placement on 2D planar surfaces, on 3D curved surfaces, and in 3D volumes.

Acknowledgements

This work was supported in part by the DoD HPCVI program, NSF grant EPS-0132618, and the NGI URI program. The authors would like to thank Abdelkrim Mebarki for his online demo. Thanks also go to the anonymous reviewers for their valuable comments.

References

[1] J. J. van Wijk, "Spot Noise: Texture Synthesis for Data Visualization," *Proc. ACM SIGGRAPH '91*, pp. 309-318, 1991.

[2] B. Cabral and L. Leedom, "Imaging Vector Fields Using Line Integral Convolution," *Proc. ACM SIGGRAPH '93*, pp. 263-270, 1993.

[3] D. Stalling and H.-C. Hege, "Fast and Resolution Independent Line Integral Convolution," *Proc. ACM SIGGRAPH '95*, pp. 249-256, 1995.

[4] H.-W. Shen and D. L. Kao, "A New Line Integral Convolution Algorithm for Visualizing Time-Varying Flow Fields," *IEEE Trans. Visualization and Computer Graphics*, vol.. 4, no. 2, pp. 98-108, 1998.

[5] Z. Liu and R. J. Moorhead II, "Accelerated Unsteady Flow Line Integral Convolution," *IEEE Trans. Visualization and Computer Graphics*, vol. 11, no. 2, pp. 113-125, 2005.

[6] B. Jobard, G. Erlebacher, and M. Y. Hussaini, "Lagrangian-Eulerian Advection of Noise and Dye Textures for Unsteady Flow Visualization," *IEEE Trans. Visualization and Computer Graphics*, vol. 8, no. 3, pp. 211-222, 2002.

[7] J. J. van Wijk, "Image Based Flow Visualization," *Proc. ACM SIGGRAPH '02*, pp. 745-754, 2002.

[8] B. Jobard and W. Lefer, "Creating Evenly-Spaced Streamlines of Arbitrary Density," *Proc. Eighth Eurographics Workshop on Visualization in Scientific Computing*, pp. 45-55, 1997.

[9] A. Mebarki, P. Alliez, and O. Devillers, "Farthest Point Seeding for Efficient Placement of Streamlines," *Proc. IEEE Visualization '05*, pp. 479-486, 2005.

[10] G. Turk and D. Banks, "Image-Guided Streamline Placement," *Proc. ACM SIGGRAPH '96*, pp. 453-460, 1996.

[11] X. Mao, Y. Hatanaka, H. Higashida, and A. Imamiya, "Image-Guided Streamline Placement on Curvilinear Grid Surfaces," *Proc. IEEE Visualization '98*, pp. 135-142, 1998.

[12] B. Jobard and W. Lefer, "Unsteady Flow Visualization by Animating Evenly-Spaced Streamlines," *Proc. Eurographics '00,* pp. 21-31, 2000.

[13] B. Jobard and W. Lefer, "Multiresolution Flow Visualization," *Proc. Ninth International Conf. in Central Europe on Computer Graphics, Visualization, and Computer Vision (WSCG '01),* pp. 33-37, 2001.

[14] O. Mattausch, T. Theubl, H. Hauser, and E. Groller, "Strategies for Interactive Exploration of 3D Flow Using Evenly-Spaced Illuminated Streamlines," *Proc. Nineteenth Spring Conf. on Computer Graphics*, pp. 213-222, 2003.

[15] V. Verma, D. Kao, and A. Pang, "A Flow-Guided Streamlines Seeding Strategy," *Proc. IEEE Visualization '00*, pp. 163-170, 2000.

[16] X. Ye, D. Kao, and A. Pang, "Strategy for Seeding 3D Streamlines," *Proc. IEEE Visualization '05*, pp. 471-478, 2005.

[17] T. Wischgoll and G. Scheuermann, "Detection and Visualization of Closed Streamlines in Planar Flows," *IEEE Trans. Visualization and Computer Graphics,* vol. 7, no. 2, pp. 165-172, 2001.

[18] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel, "Grid-Independent Detection of Closed Streamlines in 2D Vector Fields," *Proc. Ninth International Fall Workshop on Vision, Modeling, and Visualization (VMV'04)*, pp.421-428, 2004.

[19] A. Mebarki, "Demo Executable" http://www-sop.inria.fr/geometrica/team/Abdelkrim.Mebarki.

[20] D. Higham, "Highly Continuous Runge-Kutta Interpolants," *ACM Trans. Mathematical Software*, vol. 17, no. 3, pp. 368-386, 1991.

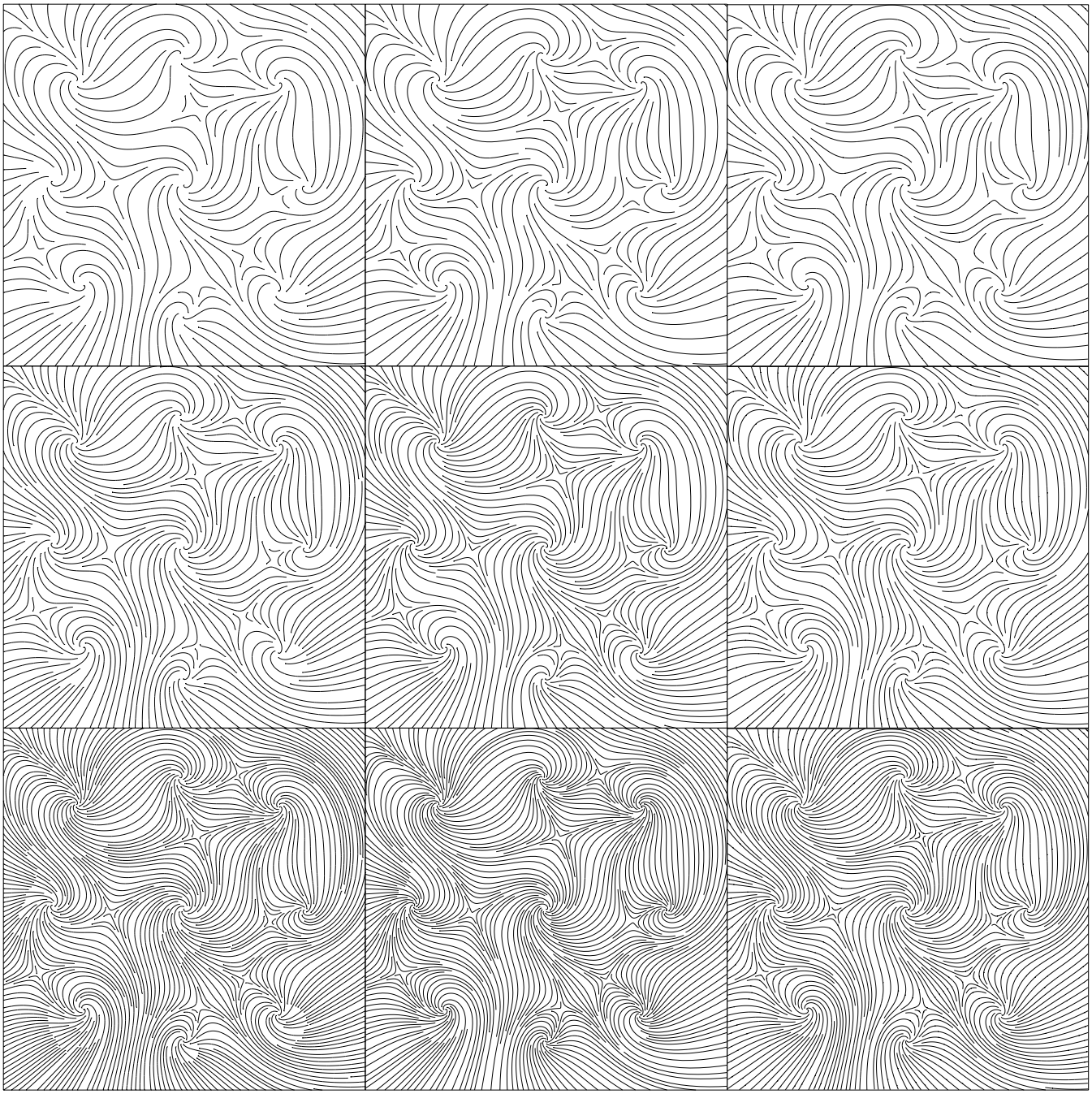

Fig. 8. Streamline placements generated by Jobard and Lefer's algorithm (left column), our ADVESS algorithm (middle column), and Mebarki et al.'s algorithm (right column) for three increasing densities (top to bottom, 2.0%, 1.5%, and 1.0%) of a 300 × 300 flow field.

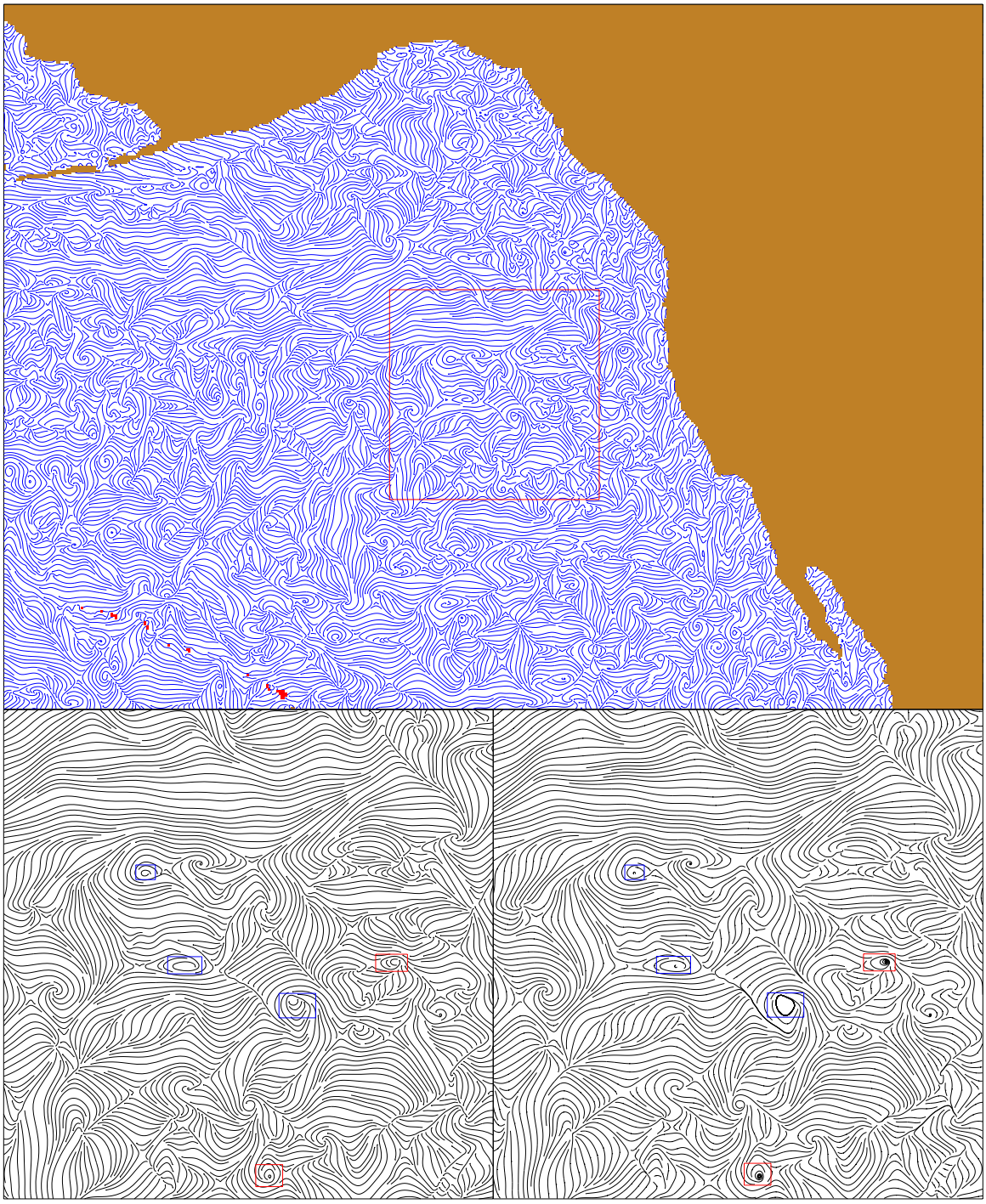

Fig. 9. Our ADVESS algorithm is effective in generating the placement (upper, separating distance: 2.0 in cell size) of evenly-spaced streamlines for a 468 × 337 flow field obtained through a US Navy model of the Northeast Pacific Ocean (180° W ~ 78° W, 20° N ~ 62° N). Also shown are the two placements (separating distance: 2.4 in cell size) generated by our algorithm (lower left, time used: 0.362 seconds) and Mebarki et al.'s algorithm (lower right, time used: 1.684 seconds) for a zoomed region (data size: 201 × 201) of the original ocean flow field. Our algorithm is more robust in loop detection and 4.6 times faster in placement generation than Mebarki et al.'s algorithm.