Parallel Streamline Placement for 2D Flow Fields

Wenyao Zhang, Yi Wang, Jianfeng Zhan, Beichen Liu, and Jianguo Ning

Abstract—Parallel streamline placement is still an open problem in flow visualization. In this paper, we propose an innovative method to place streamlines in parallel for 2D flow fields. This method is based on our proposed concept of local tracing areas (LTAs). An LTA is defined as a subdomain enclosed by streamlines and/or field borders, where the tracing of streamlines are localized. Given a flow field, it is initialized as an LTA, which is later recursively partitioned into hierarchical LTAs. Streamlines are placed within different LTAs simultaneously and independently. At the same time, to control the density of streamlines, each streamline is associated with an isolation zone and a saturation zone, both of which are center aligned with the streamline but have different widths. None of streamlines can trace into isolation zones of others. And new streamlines are only seeded within valid seeding areas (VSAs) that are enclosed by saturation zones and/or field borders. To implement the parallel strategy and the density control, a cell-based modeling is devised to describe isolation zones and LTAs as well as saturation zones and VSAs. With the help of these cell-based models, a heuristic seeding strategy is proposed to seed streamlines within irregular LTAs, and a cell-marking technique is used to control the seeding and tracing of streamlines. Test results show that the placement method can achieve highly parallel performance on shared memory systems without losing the quality of placements.

Index Terms—Flow visualization, parallel algorithms, seeding strategies, streamline placement

1 Introduction

In flow visualization, streamline is a simple and straightforward way to disclose flow patterns. Many streamline placement methods have been developed, for example, the image-guided placement [1], the neighborhood seeding method [3], the farthest point seeding method [6], the flow-guided seeding method [7], the topology-aware placement [13], the information-aware placement [16], etc. However, parallelization of streamline placement is still an open problem. There is only previous work related to parallel streamline generation [22], [23], [24], [25], [26], [34], [35], where seed points are often determined in advance without considering the interference of streamlines.

For streamline placement, the first concern is the placement quality which mainly depends on the relationships of streamlines. The parallelization must consider some basic requirements such as separating distances between streamlines. Existing parallel strategies for streamline generation in [18], [19], [22], [23], [24], [25], [26], and [35] do not take account of such important issues. When these strategies are used to place streamlines, the quality of placements is usually damaged by artificial boundaries or visual clutters, of which we defer the discussion in Section 3.

In this paper, we propose an innovative method to place streamlines in parallel, which can avoid artificial boundaries and visual clutters caused by improper parallelization. Our method is motivated by topological analysis [28], but it can run without any explicit topological analysis. The parallel processing is based on our proposed concept of *local tracing areas* (LTAs). *An LTA is a subdomain enclosed by streamlines and/or field borders, where the tracing of streamlines is completely localized*. The local tracing property of LTAs enables us to *place streamlines within different LTAs simultaneously and independently*. Furthermore, when a streamline is placed in an LTA, it will partition the LTA into sub-LTAs where we can recursively place streamlines. Therefore, initializing the whole field as an LTA, we can iteratively partition the field into a hierarchy of LTAs by placing streamlines. At the same time, to control interstreamline distances, we set an *isolation zone* and a *saturation zone* for each newly generated streamline, both of which are center aligned with the streamline but have different widths, and rule that: 1) none of streamlines can trace into isolation zones of others; 2) new streamlines are only seeded within *valid seeding areas* (VSAs) that are enclosed by saturation zones and/or field borders. LTAs enable us avoiding topological analysis. Meanwhile, introducing isolation zones and saturation zones makes it easy to control the maximum interstreamline distance as well as the minimum interstreamline distance. To describe LTAs and VSAs that may be in any shape and size, we construct two orthogonal grids covering the field, and then *model LTAs and VSAs as sets of empty cells that are four-connected in their corresponding grids*, respectively. On this basis, a heuristic seeding strategy is proposed to seed streamlines within cell-based LTAs, and a cell-marking technique is used to control the seeding and

- *W. Zhang and B. Liu are with the Beijing Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, No. 5 South Zhongguancun Street, Haidian District, Beijing 100081, P.R. China. E-mail: zhwenyao@bit.edu.cn, frostpolaris@gmail.com.*
- *Y. Wang and J. Ning are with the School of Mechatronical Engineering, Beijing Institute of Technology, No. 5 South Zhongguancun Street, Haidian District, Beijing 100081, P.R. China. E-mail: {19751115, jgning}@bit.edu.cn.*
- *J. Zhan is with the State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, No. 6 Kexueyuan South Road Zhongguancun, Haidian District, Beijing 100190, P.R. China. E-mail: zhanjianfeng@ict.ac.cn.*

Manuscript received 3 Nov. 2011; revised 10 May 2012; accepted 31 July 2012; published online 8 Aug. 2012.
Recommended for acceptance by D. Weiskopf.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2011-11-0266.
Digital Object Identifier no. 10.1109/TVCG.2012.169.

tracing of streamlines efficiently. Finally, our proposed algorithm can place streamlines in parallel, and achieve highly parallel performance on shared memory systems without losing the quality of placements.

The remainder of this paper is organized as follows: Section 2 briefly reviews related work. Section 3 states the problem of parallel streamline placement. Section 4 presents our approaches for parallel streamline placement, including the parallel strategy based on LTAs, the density control of streamlines, the cell-based models, the heuristic seeding strategy, and the cell-marking technique. Section 5 includes the full algorithm and its implementation. Test results of the algorithm are given in Section 6. The limitations, remarks, and possible future work are discussed in Section 7. Section 8 is the conclusions of the paper.

2 Related Work

Many methods have been proposed for streamline placement in previous work. Some of them aim to achieve evenly spaced streamlines, while others try to preserve or highlight flow features.

Much earlier work is the image-guided placement proposed by Turk and Banks [1]. This work can produce high quality of placements by iteratively optimizing the distribution of streamlines, but the computation is heavy. Later, Mao et al. [2] extended this work on curvilinear grid surfaces. Jobard and Lefer [3] proposed a neighborhood seeding strategy which selects seed points from candidates that have a threshold distance from an accepted streamline. When a new streamline is generated, its distance from all existing ones is checked to make it not less than the threshold. This work was later improved by Liu et al. [4] to reduce cavities and discontinuities in final placements. It was also extended for surfaces by Spencer et al. [5]. To favor long streamlines, Mebarki et al. [6] presented the farthest point seeding strategy, where Delauney triangulation is applied to seed a new streamline at the farthest point from all existing ones.

To capture flow patterns near critical points, Verma et al. [7] presented a flow-guided placement method that depends on the identification of critical points. If no any critical point is identified in the field, this approach fails. Similar work for 3D flow fields reported by Ye et al. [8] also has this problem. In addition to critical points, other flow features such as periodic orbits and separatrices [9] are also used to guide the placement of streamlines. To highlight some flow features without feature extraction, Li et al. [10] took a metric of dissimilarity to select streamlines dissimilar to all existing ones. Chen et al. [11] applied a similarity metric to control the growth of streamlines that are seeded in the field randomly and densely.

Recently, Zhang and Deng [12] presented a topology-driven placement method for evenly-spaced streamlines. Wu et al. [13] presented a topology-aware method to seed new streamlines along paths that are orthogonal to all streamlines in topological regions. Both methods utilize the topology of flow fields. When the topology is not available, the approach proposed in [12] still works as demonstrated in [14]. To capture underlying flow features, Yu et al. [15] built up a hierarchy of streamline bundles by clustering a large number of densely distributed streamlines, and obtained placements at different levels of detail. From the information communication point of view, Xu et al. [16] developed an information-aware placement method, which iteratively adds and prunes streamlines until they can convey the information contained in the original vector field.

All above methods can achieve desired placements with appropriate efforts. But, to the best of our knowledge, it is still an open problem to place streamline in parallel, while parallel computing has been successfully applied in volume rendering [17] and texture-based flow visualization [18], [19], [20], [21] as an efficient way to deal with large data sets. There are only several parallel strategies devised for streamline generation.

In [22], Sujudi and Haimes introduced a parallel strategy to generate streamlines, where a flow field is partitioned into rectangular blocks which are then assigned to different processors to compute streamlines. For this strategy, streamlines are allowed to travel across blocks. The communication of streamlines between processors will significantly degrade the parallel performance.

To reduce communication overhead and keep load balancing, Chen and Fujishiro [23] presented an irregular data repartitioning scheme based on spectral graph decomposition, and Yu et al. [24] presented an irregular domain partition scheme based on hierarchical clustering of the vector field. These two methods are designed for streamline generation, not suitable for parallel streamline placement, because visual clutters will occur in results if streamlines are allowed to travel across decomposed subdomains.

Pugmire et al. [25] further investigated several parallel strategies for generating densely distributed streamlines where regular domain decompositions are used. Peterka et al. [26] showed how to apply and scale parallel particle tracing to very large systems. Nouanesengsy et al. [34] achieved load-balanced streamline generation by decomposing the flow field into blocks with nearly equal workloads. Hlawatsch et al. [35] proposed an acceleration scheme to generate streamlines, where initial streamlines are iteratively prolonged in parallel until a designated length is achieved. In these methods, the seeding of streamlines is not concerned. They cannot be directly used for parallel streamline placement.

Other related work for parallel streamline generation is associated with the technique of line integral convolution which is widely used to compute textures of flow fields in [18], [19], [20], and [21].

3 The Parallelism of Streamline Placement

Streamlines are integral curves tangent to the underlying flow field. Given a seed point, we can generate a streamline by an integration method such as Euler or Runge-Kutta [27]. If seeds are determined beforehand and relations of streamlines are not concerned, we can generate streamlines according to some parallel processing strategies [22], [25], and achieve highly parallel performance.

For streamline placement, however, seed points are not determined in advance. We need to select seed points and then generate streamlines. At the same time, we also need to control the distribution of streamlines to meet some

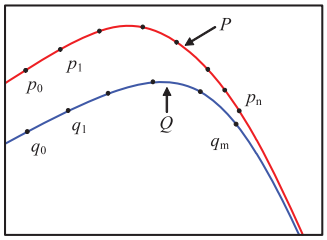


Fig. 1. Illustration of two streamlines computed in parallel.

requirements such as uniformity and feature preservation [13]. As a result, when seeding and generating a new streamline, we must consider its effect on all existing ones. This makes it difficult to place streamlines in parallel. In the rest of this section, we will state the challenges of parallel streamline placement for possible cases.

If we place two streamlines in a field simultaneously without any synchronization, the separating distance of streamlines would be out of control. For example, as shown in Fig. 1, suppose that there are two streamlines in computing, which are named P and Q, respectively, and let points p_n and q_m be the next samples that are being computed independently. To ensure the separating distance between P and Q is not less than a threshold, we must check the distance from p_n to q_m, and vice versa. Obviously, this checking has to be delayed until both p_n and q_m are determined. It means that we need to synchronize the integration of streamlines. The synchronization will degrade the performance of parallel processing severely. In some special cases, the parallel placement may even degenerate into a sequential one with extra overhead. When placing multiple streamlines in a field concurrently, we have the same problems, but more complicated.

As an alternative, we may choose to place streamlines in different subdomains simultaneously. This is in fact a parallel strategy based on domain decompositions, where flow fields are generally partitioned into regular blocks which are then assigned to different processors placing streamlines. If streamlines are not allowed to travel across blocks, placements in different blocks are completely independent. We can achieve highly parallel performance without any communications among processors. The disadvantage is that the final placements will include some *artificial boundaries* caused by discontinuities of streamlines at blocks' borders. Such an example is shown in Fig. 2, where the field domain is equally partitioned into four rectangular blocks. In Fig. 2, we can clearly perceive the artificial boundaries indicated by discontinuities of streamlines. These artificial boundaries are not real contents of the field. They not only impair the placement, but also interfere with the understanding of the flow field.

To overcome artificial boundaries, we must allow streamlines to travel across blocks. In this case, there are two possible approaches to compute streamlines involving two or more blocks. The first approach is to compute a streamline always using its originally assigned processor. By this way, two streamlines within a block may be computed by two different processors, just like placing two streamlines in a field simultaneously. As explained in the early part of this section, it is not feasible. The second approach is to communicate streamlines among processors. When a streamline leaves a block and enters a new one, its

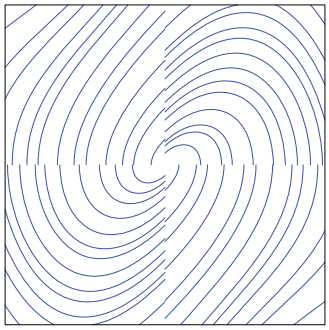


Fig. 2. An example of artificial boundaries. The field domain is equally partitioned into four rectangular blocks, each of which is assigned to a processor placing streamlines independently. Streamlines are not allowed to travel across blocks.

tracing is transferred to the processor in charge of the new block. By this means, streamlines may enter a block at any time and any place. If the entrance is allowed without any control, visual cluttering will inevitably occur in the resulting placement. Such an example is shown in Fig. 3, where all settings are the same as those in Fig. 2, except that the streamlines are allowed to travel across blocks. Although artificial boundaries are avoided, the placement quality in Fig. 3 is not improved due to the visual clutters caused by parallel processing. If we try to control the entrance of streamlines, we must synchronize the seeding and generating of streamlines among processors. The synchronization will not only complicate the placement method, but also hurt the parallel performance. In some special cases, it may make the parallel processing degenerate into a sequential one.

The above analyses show that regular domain decompositions are not suitable for parallel streamline placement though they are widely used in parallel streamline generation. Besides, there are several irregular decompositions that have been proposed in [23], [24], and [25] with the goal of minimizing communications required by parallel streamline generation. When these irregular decompositions are used for parallel streamline placement, we still have the same problems as discussed for regular decompositions, due to the fact that streamlines inevitably travel across decomposed subdomains.

Therefore, we conclude that it is still an open problem to place streamlines in parallel without sacrificing the quality of placements. In the next section, we will provide a solution for this challenging problem.

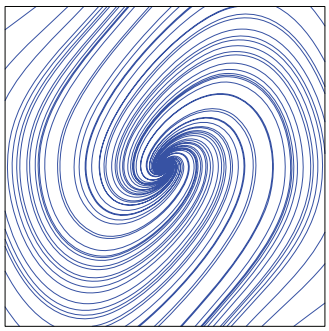
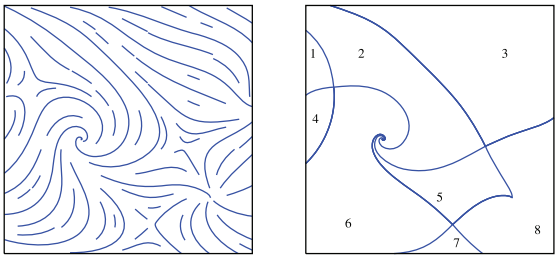


Fig. 3. Example of visual clutters caused by improperly placed streamlines. All settings are the same as those in Fig. 2, except that streamlines are allowed to travel across blocks.



(a) (b)

Fig. 4. An example of topological partition. The field, which is visualized by streamlines in (a), is partitioned into eight topological areas by its topological skeleton in (b).

4 Our Approach

This section presents our approach for parallel streamline placement, including the parallel strategy based on an irregular domain decomposition driven by streamlines, the density control of streamlines, the seeding and tracing of streamlines, as well the cell-based models used to implement the parallel strategy and the density control.

4.1 Parallel Strategy

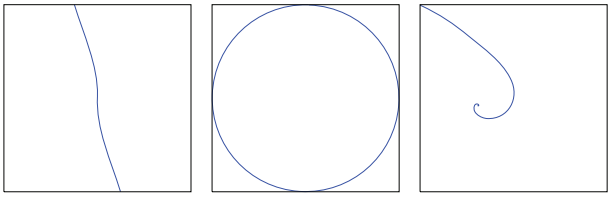
Our approach for parallel streamline placement is motivated by topology analysis of flow fields [28]. As shown in Fig. 4, a flow field can be partitioned into topological areas by its topological skeleton. Any streamline seeded in a topological area cannot travel into other areas. Its tracing is completely restricted within its local area. This local property is very important for parallel processing. We refer to it as the *local tracing property of streamlines* in this paper. Moreover, when a streamline is placed in a topological area, it often partitions the area into two small ones that still have the local tracing property of streamlines.

Therefore, we can *place streamlines within different topological areas simultaneously and independently*. Obviously, this parallel strategy is based on an irregular domain decomposition that is driven by flow behaviors. The issues of artificial boundaries and visual clutters, as discussed in Section 3, can be theoretically overcome because of the local tracing property of streamlines within topological areas. In practice, however, it is not easy to get topological partitions of flow fields due to the complexity of topology analysis.

To avoid topological analysis, we introduce a concept of *local tracing areas*, and develop a hierarchical domain decomposition based on LTAs.

Given a flow field, *an LTA is defined as a subdomain enclosed by streamlines and/or field borders*. By this definition, LTAs also have the local tracing property of streamlines, because any streamline seeded in an LTA cannot travel into other LTAs. In this sense, LTAs are similar to topological areas, but LTAs are technically wider than topological areas. An LTA can be a full topological area, a part of a topological area isolated by streamlines, or a set of topological areas.

According to the definition, a flow field can be initialized as an LTA. When a streamline is placed in an LTA, it may partition the LTA into *one or more* sub-LTAs. Such examples are given in Fig. 5. For the first example in Fig. 5a, the streamline segments the initial LTA into two parts. Each of them is still a valid LTA, so they are referred



(a) (b) (c)

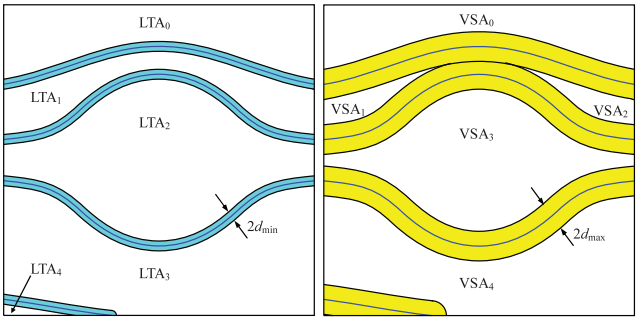
Fig. 5. Examples of sub-LTAs. The initial LTAs standing for flow fields are partitioned into (a) two sub-LTAs, (b) five sub-LTAs, and (c) only one sub-LTA, respectively.

to as *sub-LTAs*. For the second example in Fig. 5b, the initial LTA standing for the field is partitioned into five sub-LTAs by the circular streamline which is tangent to the field borders. In Fig. 5c, the streamline does not segment the initial LTA, but change it into a new one, which is also a sub-LTA. No matter what happened, newly generated LTAs will still have the local tracing property of streamlines. Therefore, *by initializing a flow field as an LTA, we can recursively partition the flow field into hierarchical LTAs, and place streamlines within different LTAs independently and simultaneously without any topological analysis*. This is the basic idea of our parallel streamline placement.

4.2 Density Control

One basic goal of a placement method is to make streamlines as uniformly spaced as possible. The uniformity of streamlines can be measured by two parameters: the minimum interstreamline distance, $d_{\min}$, and the maximum interstreamline distance, $d_{\max}$. To get uniform placements, we must set these two parameters with proper values, and ensure that the separating distance of streamlines is neither lower than the minimum nor higher than the maximum.

The minimum distance requirement can be equivalently expressed as two rules: 1) Each streamline has an *isolation zone*, which is center aligned with the streamline and has a fixed width equal to $2d_{\min}$, just as shown in Fig. 6a, where isolation zones associated with streamlines are indicated by light blue ribbons; 2) No any streamline can trace into other streamlines' isolation zones. If we can maintain isolation zones for all streamlines, the minimum distance requirement can be ensured. In this case, *LTAs are changed into*



(a) (b)

Fig. 6. Isolation zones and saturation zones set for streamlines. (a) shows isolation zones whose width is $2d_{\min}$, (b) shows saturation zones whose width is $2d_{\max}$.

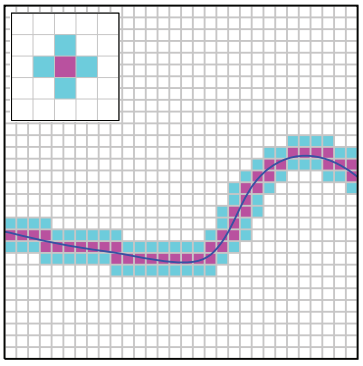


Fig. 7. An example of cell-based isolation zone. The isolation zone of the blue streamline is represented by the visited cells (in magenta) and the forbidden cells (in cyan).

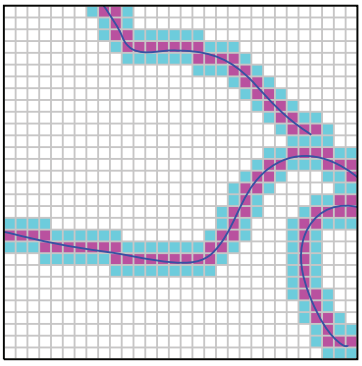


Fig. 8. Examples of cell-based LTAs and isolation zones.

subdomains enclosed by isolation zones and/or field borders. The local tracing property of streamlines is not influenced. We can still place streamlines in parallel within LTAs, and set isolation zones for newly placed streamlines.

For the maximum distance requirement, we can also set a zone for each streamline that is similar to the isolation zone but has a different width equal to $2d_{\max}$. Here, we refer to such zones as *saturation zones*. As shown in Fig. 6b, saturation zones will also segment the field into subdomains. To meet the maximum distance requirement economically, we should only seed new streamlines in these subdomains. For this reason, *we refer to such subdomains enclosed by saturation zones and/or field borders as valid seeding areas*. Correspondingly, *saturation zones are subdomains without the need to seed new streamlines*. If the field domain is completely covered by saturation zones, the placement will stop at a desired density of streamlines.

Because saturation zones are wider than isolation zones, the relationship of VSAs and LTAs can be summarized as: 1) Any VSA is completely embedded in an LTA; 2) An LTA can contain one or more VSAs, or just be empty without any VSA. Such examples are given in Fig. 6, where LTA_0 contains VSA_0, and LTA_1 contains VSA_1 and VSA_2, while LTA_4 does not include any VSA.

4.3 Cell-Based Models

The parallel strategy and the density control are straightforward. The key is how to implement them efficiently, where two problems must be solved. One is to describe LTAs and VSAs that may be in any shape and size. The other is to maintain isolation zones and saturation zones for all streamlines. If we deal with them based on the distance control of samples, the cost of computation will be very high. In this section, we propose a cell-based mechanism to model isolation zones and LTAs, and then extend it to saturation zones and VSAs.

4.3.1 Modeling for Isolation Zones and LTAs

Isolation zones and LTAs are both subdomains in flow fields. The difference is that isolation zones' shapes are determined by streamlines, while LTAs may be in any shape and size. To describe them in an efficient way, we construct an orthogonal control grid to cover the field domain with *empty cells*, just as shown in Fig. 7. *When a streamline is placed in the field, we render it in the control grid by marking all cells that are visited by the streamline*. By this means, a streamline can be represented as a set of marked cells in the grid. For the streamline in Fig. 7, the visited cells are marked in magenta. This *cell-marking technique* is similar to the cell-based counting scheme used in Fast LIC [29]. The visited cells can describe the streamline in the tracing grid. But they are not enough to control the minimum interstreamline distance, because two streamlines located in two adjoining cells can approach to each other infinitely. In order to isolate streamlines effectively, we continue marking the cells that are the direct neighbors of the visited cells in terms of four connection, and forbid other streamlines to enter these cells. That is to say, if a cell is marked as *visited* by a streamline, its left and right neighbors will be marked as *forbidden* as well as its upper and lower ones, as shown in the top left close-up of Fig. 7, where the visited cell is drawn in magenta and the forbidden cells are drawn in cyan. Here, it should be noted that when a cell is marked as visited, it will not be remarked as forbidden. *All of these marked cells, including the visited and the forbidden, are used to model the isolation zone of the streamline*. According to the definition of isolation zones, other streamlines are not allowed to enter into all marked cells. In this sense, we refer to the underlying grid as the *tracing grid* for its control for the tracing of streamlines. Given a flow field, we can set the tracing grid with desired cell sizes. For simplicity, we suppose that each cell is a square with a size of λ.

When isolation zones are determined, the tracing grid will be isolated into *empty four-connected regions*. For example, the tracing grid in Fig. 7 is isolated into two disjoint parts, each of which is a set of empty cells that are four connected. When streamlines are seeded in such regions, their tracing will be confined within their local regions, because marked cells are not open for other streamlines. In this sense, *an empty four-connected region in the tracing grid is an LTA (a cell-based LTA)*. Therefore, we can model an LTA approximately as a set of empty cells that are four connected in the tracing grid. More examples for cell-based LTAs are given in Fig. 8, where three streamlines segment the field into four LTAs.

4.3.2 Modeling for Saturation Zones and VSAs

Saturation zones are formally similar to isolation zones, except that the former are wider than the latter. We can use the tracing grid to model saturation zones like isolation

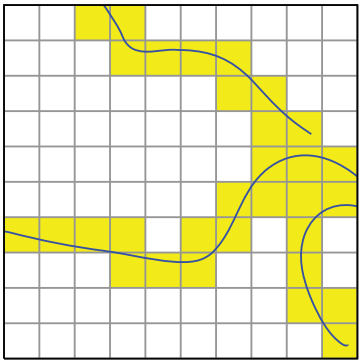


Fig. 9. Examples of cell-based saturation zones and VSAs. A streamline's saturation zone is represented as the cells visited by the streamline in the grid.

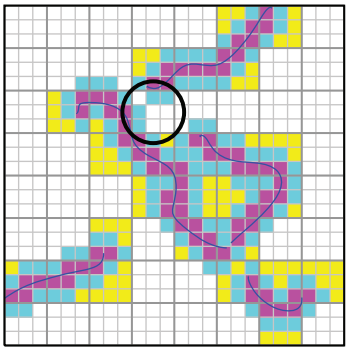


Fig. 10. The mapping between the tracing grid and the seeding grid. Seeding cells are drawn in thick lines. Tracing cells are drawn in thin lines. The saturated seeding cells are shown in yellow. The visited and the forbidden tracing cells are shown in magenta and cyan, respectively.

zones. In terms of functionality, however, saturation zones are different from isolation zones. Saturation zones are mainly used to help seed streamlines to meet with the maximum distance requirement. For the sake of clarity, we construct another orthogonal control grid to model saturation zones, and name it as the *seeding grid* for its role in the control of seeding streamlines. When a streamline is placed in the field, we *set its saturation zone as a set of cells that are visited by the streamline in the seeding grid*, just as shown in Fig. 9, where the saturation zones of the blue streamlines are represented as the yellow cells visited by the streamlines. Correspondingly, a VSA is approximately modeled as a set of empty cells that are four connected in the seeding grid, i.e., *an empty four-connected region in the seeding grid is treated as a VSA (a cell-based VSA)*. There are four cell-based VSAs in Fig. 9. Given a flow field, we can set the seeding grid with desired cell sizes. For simplicity, we suppose that all cells in the seeding grid are squares with a size of γ.

In the cell-based modeling of saturation zones, we do not include the neighbors of the visited cells as done for isolation zones. The reason is that the maximum distance requirement is actually achieved by VSAs instead of saturation zones. We only take saturation zones to find out all VSAs to seed new streamlines. The problem in maintaining interstreamline distances does not happen to saturation zones. The cells visited by streamlines are enough to model saturation zones in the seeding grid.

4.3.3 Relations of Cell-Based LTAs and VSAs

In theory, VSAs are completely embedded in LTAs. This relationship may be broken when the cell-based models are used for LTAs and VSAs. To correct the broken relationship, we refer to cells of the tracing grid as *tracing cells*, and those of the seeding grid as *seeding cells*. Moreover, we set $\gamma = n * \lambda$, where n is an integer not less than three. With these settings, we superpose the seeding grid to the tracing grid, as shown in Fig. 10, where seeding cells and tracing cells are drawn in thick and thin lines, respectively. By reviewing tracing cells and seeding cells together, we can obtain a mapping between the two grids as follows:

1. Each tracing cell is within a seeding cell, and each seeding cell contains multiple tracing cells. Given a tracing cell, we can easily get its related seeding cell, and vice versa.
2. If a tracing cell is marked as *visited* (in magenta), the related seeding cell must be marked as *saturated* (in yellow); if a seeding cell is marked as saturated, one of its tracing cells must be marked as visited.
3. If a seeding cell is unmarked, none of its tracing cells can be mark as visited, and only the tracing cells along its borders can be marked as *forbidden* (in cyan).

According to the mapping, most of unmarked seeding cells will be contained in a cell-based LTA (i.e., an empty four-connected region in the tracing grid), except for a special case where the unmarked seeding cell is divided into disjoint parts by forbidden tracing cells on its borders. An example for the special case is shown in the circle of Fig. 10, where the seeding cell is divided into two parts that belong to different LTAs. In this case, we rule that *the seeding cell only belongs to the LTA that includes the center tracing cell of the seeding cell*. This rule enables us to maintain the embedded relationship of VSAs and LTAs. We can search an LTA in the tracing grid, and get its possible associated VSAs from the seeding grid at the same time without any ambiguity. Here, we should note that: 1) not all LTAs have VSAs, and 2) LTAs may be small enough to hold only one tracing cell. Streamlines will be placed in LTAs that have VSAs regardless of LTAs' size. LTAs without any VSA will be discarded simply. In Fig. 10, the top-left empty tracing cell in the circle is a LTA without any VSA according to the rule.

4.4 Seeding Strategy

According to the density control, we should iteratively seed streamlines in VSAs that are embedded in LTAs. For a specific LTA, if it contains one or more VSAs, any point in its VSAs is a theoretically valid seed that will not violate the density control. For the purpose of parallel processing, however, the ideal choice should be that the newly placed streamline segments the LTA into equal parts that can be processed with equal efforts. But, in practice, it is not easy to get the ideal result, because the LTA may be in any shape and size.

As an alternative, we choose to seed a new streamline at the centroid of the biggest VSA of the LTA. This seeding strategy is based on a straightforward observation. That is, if the LTA is a topological area and only one streamline is allowed to link its source and sink points, then the streamline through the centroid of the area may be the

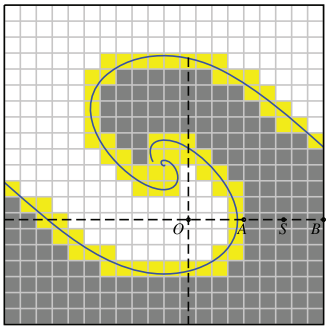


Fig. 11. Seed selection for a concave VSA. For the VSA in gray, the midpoint S of the line segment AB is used as a valid seed, since the centroid O is out of the VSA.

longest one that most likely at the same time segments the LTA into two roughly equal parts. Although the observation is not always true due to the complexity of LTAs, this heuristic seeding strategy does work in practice, just as demonstrated in Section 6.

When cell-based models are used for LTAs and VSAs, our seeding strategy works as followed. Given an LTA, we pick out the biggest VSA to compute its centroid by simply averaging the center coordinates of the cells contained in the VSA. If the centroid lies in the VSA, we select it as a seed point; otherwise, we draw two straight lines through the centroid in vertical and horizontal directions, respectively, and choose the midpoint of the longest line segment intercepted by the VSA as a seed. For the latter case, an example is shown in Fig. 11, where the VSA in gray is a concave polygon, and its centroid O is outside, so the midpoint S of the line segment AB is used as a seed. The processing shown in Fig. 11 is a correction for the exception of the centroid-based seeding method, which can be implemented without too much overhead. Other similar measures can be used in practical applications. Moreover, if the selected seed point is in a marked tracing cell, it will be replaced with the center of its affiliated seeding cell to make sure it is a valid seed. Finally, if an LTA does not contain any VSA, it will be skipped without placing any streamline.

4.5 Streamline Generation

When the seed point is determined for an LTA, we can generate the streamline starting from the seed with an integration method such as Euler or Runge-Kutta [27]. In this paper, fourth-order Runge-Kutta method is applied with a fixed step size. The integration of streamlines will end at the reach of critical points, field borders, or isolation zones (i.e., marked cells in the tracing grid). At the same time, isolation zones and saturation zones are set for newly generated streamlines according to the methods in Sections 4.3.1 and 4.3.2, respectively. If the field domain is completely covered by saturation zones (i.e., all cells in the seeding grid are marked as saturated), we will stop seeding new streamlines, and finally get a placement with the desired density of streamlines.

5 Full Algorithm

With the density control in Section 4.2 and the cell-based models in Section 4.3, the full algorithm of our parallel streamline placement can be outlined as follows:

1. Initialization
   a. Construct the tracing grid G_t and the seeding grid G_s for a given flow field, using predefined λ and γ, where $\gamma = n * \lambda$ and n is an integer not less than three.
   b. Create an initial LTA, let it be LTA_0, set $LTA_0 = G_t$, and let G_s be the unique VSA of LTA_0.
   c. Create a task queue.
   d. Create a task with LTA_0, and put it into the queue.
2. For each task in the queue, in parallel do:
   a. Let LTA_i be the LTA associated with the task.
   b. Select a seed point from the biggest VSA of LTA_i according to the seeding strategy proposed in Section 4.4.
   c. Generate a streamline using the selected seed, and set the streamline's isolation and saturation zones as addressed in Sections 4.3.1 and 4.3.2, respectively.
   d. Find out all sub-LTAs contained in LTA_i as well as their possible VSAs.
   e. Create a new task for each sub-LTA that contains one or more VSAs, and put them into the task queue.
3. Repeat Step 2 until the task queue is empty.
4. End the placement of streamlines.

In the algorithm, a single task is initially created for the initial LTA (i.e., LTA_0), which is the root of hierarchical partitioning of the whole field. When an LTA is partitioned into sub-LTAs by a streamline, new tasks are generated to continue the placement in sub-LTAs. If an LTA or sub-LTA does not contain any VSA, then no more streamlines are required by it and no task is created for it. When there is no LTA containing any VSA, i.e., the task queue is empty, the placement of streamlines ends with the desired density.

In theory, the algorithm can run on any parallel systems with multiple *processing elements* (PEs). Here, PEs can be CPUs, physical cores, or similar computing devices. In practice, however, shared memory systems are preferred, where the flow field, the tracing grid, and the seeding grid can be shared among PEs without any conflict of read-write in concurrent accesses.

On shared memory systems, PEs' processing is based on the task queue. New tasks created for new LTAs are first put into the task queue, waiting for being executed by a PE. When a PE is idle, it queries the task queue for a new task, and turns into busy by executing a new task. All PEs will keep busy as long as the task queue is not empty. We need to synchronize the task queue among PEs. But the synchronization is very small in comparison with the computation of streamlines.

If only one PE is available, the parallel algorithm will become a sequential one where streamlines are placed one by one.

6 Test Results

We have implemented and tested our algorithm on shared memory systems with multicore CPUs. The test data sets include the small fields provided by Turk and Banks [30] and Mebarki [31], the enlarged version of the small fields,

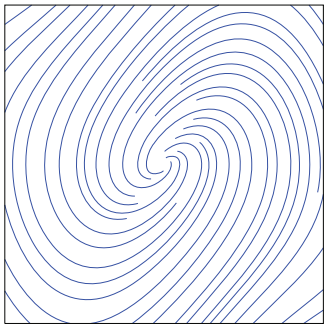


Fig. 12. The placement obtained by our parallel algorithm for the field used in Figs. 2 and 3. Four PEs are used. The field domain is $[0, 40] \times [0, 40]$, and the placement parameters are: $\gamma = 2$ and $\lambda = 0.4$.

and several complex fields synthesized by the method in [32]. Due to the limitation of paper length, this section only includes some typical results in terms of placement quality and parallel performance.

6.1 Quality of Placements

For streamline placement, quality is the first concern. There are several criteria to evaluate placement quality [13]. Here, we mainly focus on discontinuity and uniformity, since our basic goal is to make streamlines uniformly spaced as possible.

6.1.1 Artificial Boundaries and Visual Clutters

Contributing to the parallel strategy based on LTAs, our parallel algorithm can avoid artificial boundaries and visual clutters caused by improper parallelization. For the test field used in Figs. 2 and 3, the placement obtained by our algorithm is shown in Fig. 12, where neither artificial boundaries nor visual clutters can be perceived. So our method of parallel streamline placemen is successful, which is further proved by the examples in Figs. 13, 14, 15, and 16.

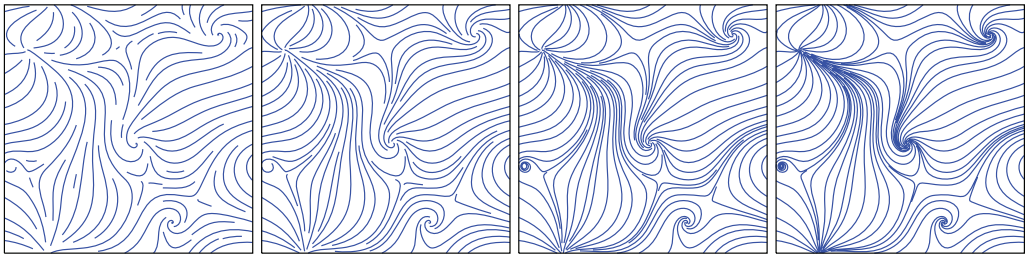
6.1.2 Uniformity of Streamlines

There are two parameters in our algorithm: λ and γ. To maintain the embedded relationship of VSAs and LTAs, we should let $\gamma = n * \lambda$ where n is an integer not less than three. Under this constraint condition, we can adjust λ and γ to achieve placements in different styles. For example, we obtain the placements in Fig. 13 by varying λ while keeping γ unchanged. It can be seen that λ mainly affects the length of streamlines. If we want to reduce short streamlines, set λ to a lower value. When short streamlines are reduced, however, the uniformity of streamlines may be broken to some extent. The placements in Fig. 14 are obtained by varying γ while keeping λ unchanged, from which we can see the density of streamlines is mainly controlled by γ. The smaller the γ is, the higher the density.

As a whole, the uniformity of streamlines is jointly determined by λ and γ. If the difference between λ and γ is small, e.g., $\gamma = 3\lambda$, the disparity of density will be small. For instance, the placements in Figs. 13a and 14e are more uniform than others. However, less disparity of density does not necessarily mean higher quality of placements, because the number of short streamlines will increase when λ is close to γ. In practice, both $\gamma = 4\lambda$ and $\gamma = 5\lambda$ are proper choices that can make a tradeoff between the uniformity and the length of streamlines.

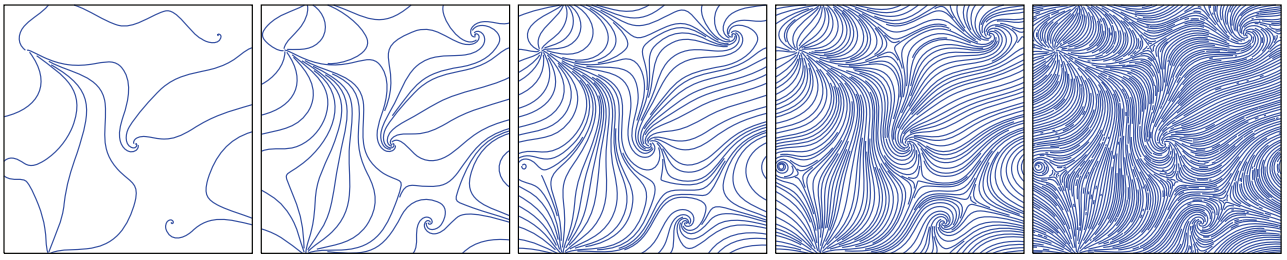
6.1.3 Comparisons with the Farthest Point Seeding

As reported in [6], the farthest point seeding method proposed by Mebarki et al. not only produces high quality of placements, but also runs faster than the methods proposed in [1] and [3]. So we select it as the counterpart compared with our method, and implement it on our test



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

Fig. 13. Placements obtained by varying λ. The field domain is $[0, 63] \times [0, 63]$. Parameter settings: (a) $\gamma = 3$, $\lambda = 1$; (b) $\gamma = 3$, $\lambda = 0.5$; (c) $\gamma = 3$, $\lambda = 0.25$; (d) $\gamma = 3$, $\lambda = 0.125$.



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

Fig. 14. Placements obtained by varying γ. The field domain is $[0, 69] \times [0, 69]$. Parameter settings: (a) $\gamma = 11.5$, $\lambda = 0.23$; (b) $\gamma = 5.75$, $\lambda = 0.23$; (c) $\gamma = 2.76$, $\lambda = 0.23$; (d) $\gamma = 1.38$, $\lambda = 0.23$; (e) $\gamma = 0.69$, $\lambda = 0.23$.

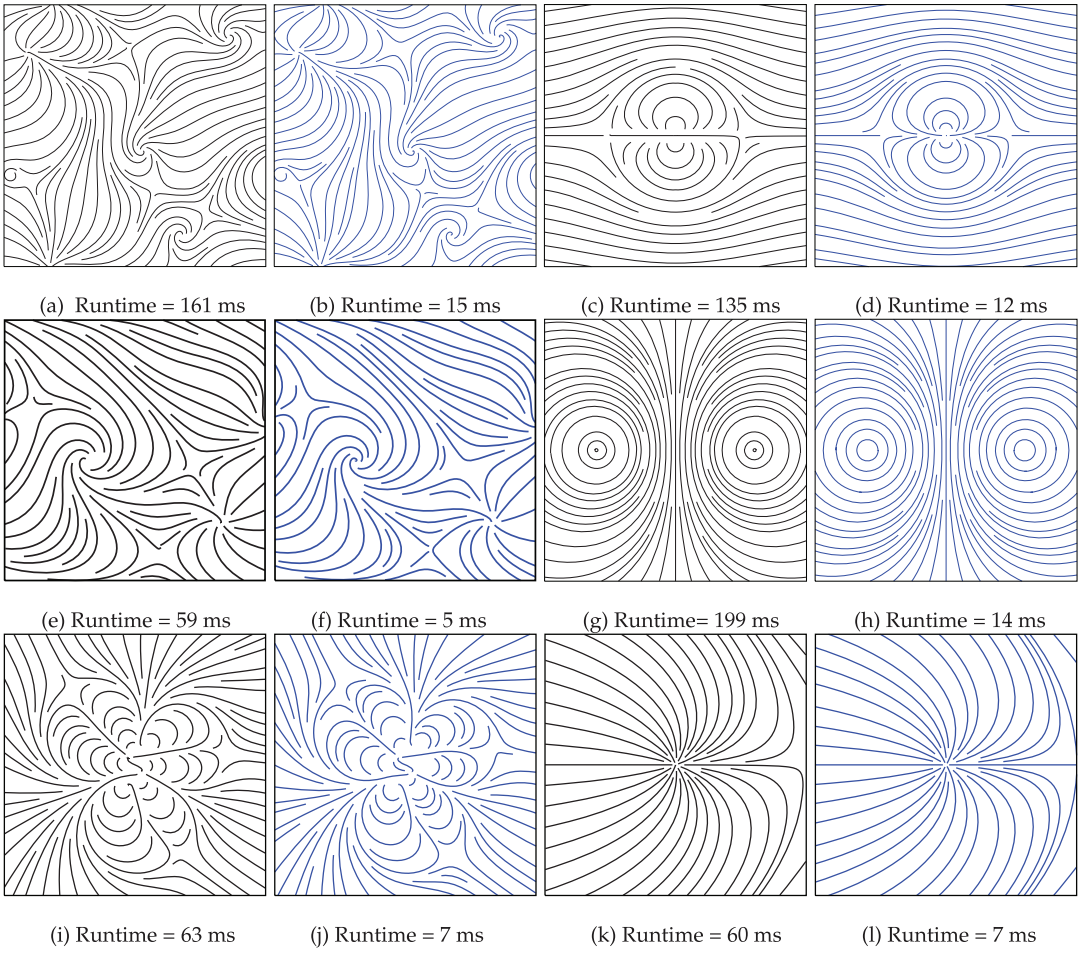


Fig. 15. Comparisons of our method with the farthest point seeding. The placements in (b), (d), (f), (h), (j), and (l) are produced by our method. The others are generated by the farthest point seeding method. For each test field, both methods are tuned to place the same number of streamlines. The runtime of each placement is labeled individually.

platform with the help of its open source code [33]. Without loss of the fairness, we run our algorithm in serial like the farthest point seeding method.

Fig. 15 depicts several comparative results, where the placements produced by our method are drawn in blue, while those produced by the farthest point seeding method are in black. The runtime of each placement is labeled individually in Fig. 15. In order to set up a basis of comparison, both methods are carefully tuned to place the same number of streamlines for each test field. Moverover, the subsampling factor of the farthest point seeding method is set to 2 for all examples. In terms of quality, our method produces comparable results with the farthest point seeding method, as shown in Fig. 15. In terms of runtime, our method is about one order of magnitude faster. For example, the placement in Fig. 15b only costs 15 milliseconds (ms) while its counterpart takes 161 ms.

6.1.4 Tests for Complex Fields

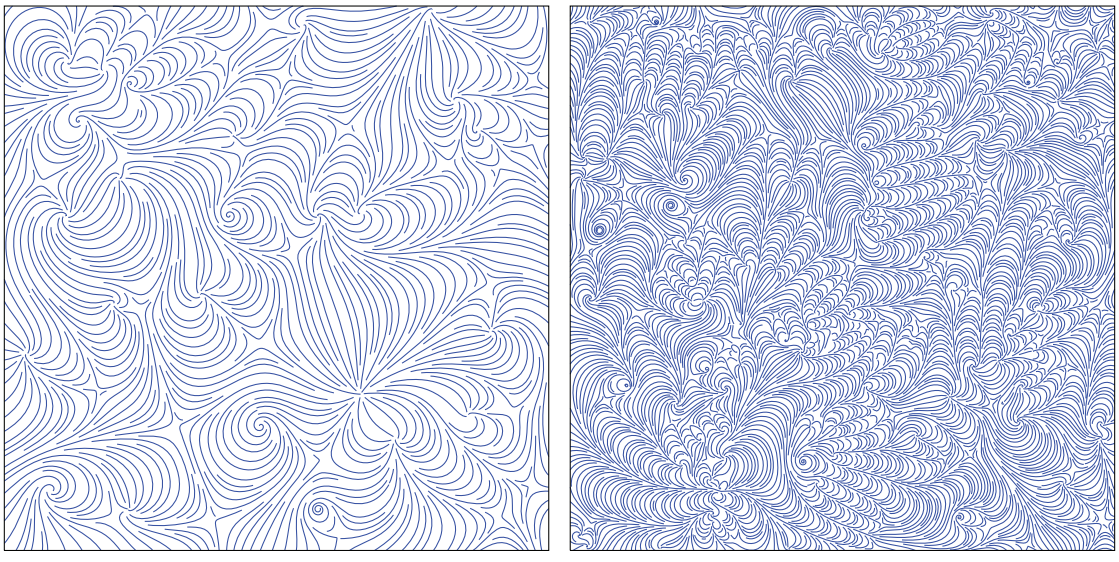
Our method is further tested with complex fields that are synthesized according to the rules of the Clifford algebra [32]. Two typical results in moderate scales are given in Fig. 16, where streamlines are as evenly spaced as possible. For the first field in Fig. 16a, it contains 25 saddles and 25 nonsaddle singularities which are randomly distributed in a domain of $[0, 512] \times [0, 512]$. The second field in Fig. 16b contains 100 saddles and 100 nonsaddle singularities which are randomly distributed in a domain of $[0, 1024] \times [0, 1024]$. Both placements are obtained with parameter settings: $\gamma = 8$ and $\lambda = 2$. For more complicated fields in larger scales, we also get similar results.

6.2 Parallel Performance

6.2.1 Runtime and Speedup

We choose two metrics to measure the parallel performance of our algorithm. One is runtime, T_p, which is defined as the elapsed time on p PEs that begins with the start of the first PE's computation and terminates at the end of the last PE's computation. The other is speedup, S_p, which is often defined as the ratio of T_1 and T_p. To measure runtime and speedup, we run our algorithm on a shared memory system equipped with 8 Intel Xeon X7550 2.00 GHz CPUs and 256 GB physical memory. Each CPU has eight physical cores. The total available PEs is up to 64.

As discussed in Section 5, the algorithm places streamlines within different LTAs simultaneously and independently. If currently available LTAs are not less than available PEs, all PEs will be busy; otherwise, some PEs will be idle wasting the power of computation. To keep more PEs busy and improve runtime, we should generate LTAs as soon as possible. The generation of LTAs depends



(a) (b)

Fig. 16. Placements of streamlines for two complex flow fields. The field in (a) contains 25 saddles and 25 nonsaddle singularities. The field in (b) contains 100 saddles and 100 nonsaddle singularities. Both placements are obtained with parameter settings: $\gamma = 8$ and $\lambda = 2$.

on the hierarchical decomposition of the flow field. If each streamline can segment its affiliated LTA into two or more equal sub-LTAs, then new LTAs will increase exponentially. This is an ideal case where we can obtain highly improved runtime. On the other hand, if there is only one available LTA to keep only one PE busy all the time, the parallel algorithm will degenerate to a sequential one without any improvement of runtime. This is the worst case. Therefore, *the parallel performance depends on the hierarchical structure of all involved LTAs*. If the structure is a balanced tree, we can achieve the best performance. If the structure is a single linked list, the worst case occurs. In practice, the parallel performance varies between the two extremes.

To show the best performance, we test our algorithm with a constant flow field whose hierarchical decomposition is a balanced binary tree. The field domain is $[0, 2520] \times [0, 2520]$. The placement settings are $\gamma = 1$ and $\lambda = 0.2$. From the curves of runtime and speedup shown in Fig. 17, we can see that the runtime is improved significantly and the speedup almost linearly grows with the increasing number of PEs. In Fig. 17b, *the fluctuation of speedup*, which also occurs in other examples, is mainly due to the deviation in measurement and the uncertainty of task scheduling in operating systems.

For some complex flow fields, results are not so perfect. For instance, we test our algorithm with the flow field shown in Fig. 16b, using the placement settings $\gamma = 1$ and $\lambda = 0.2$. The measured runtime and speedup are given in Figs. 18a and 18b, respectively, where the benefit of parallel running is rather limited. The results in Fig. 18 can be improved by supplying the algorithm with a *topological initialization*, which replaces the initial LTA with a set of real topological areas obtained by topological analysis [28]. The improved results are shown in Fig. 19. In this case, the structure of the flow field is actually known, so the algorithm can partition

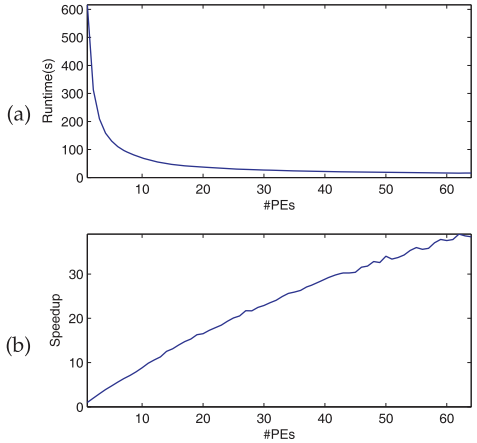


Fig. 17. The runtime and speedup obtained from a constant flow field whose domain is $[0, 2520] \times [0, 2520]$. The placement settings are: $\gamma = 1$, and $\lambda = 0.2$. (a) is the runtime versus the number of PEs. (b) is the corresponding speedup.

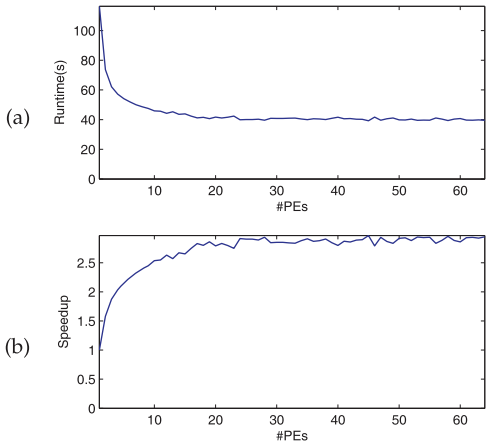


Fig. 18. The runtime and speedup obtained from the flow field shown in Fig. 16b. The field domain is $[0, 1024] \times [0, 1024]$. The placement settings are: $\gamma = 1$, and $\lambda = 0.2$. The runtime (a) and speedup (b) are drawn versus the number of PEs.

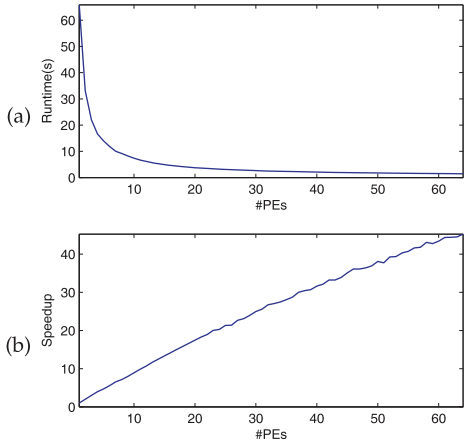


Fig. 19. The runtime and speedup improved by topological initialization. The field and placement settings are the same as those used for Fig. 18.

the field efficiently, which enables us to accelerate the placement of streamlines by increasing PEs. Compared with Fig. 18b, the speedup in Fig. 19b has been improved significantly.

For flow fields that are neither constant nor too complicated, the relationship of speedup and number of PEs is generally *a logarithmic curve*, just as shown in Fig. 20a, where the test field is an enlarged version of that shown in Fig. 14. When topological initialization is applied to the field, the speedup is also improved as shown in Fig. 20b.

6.2.2 Scalability

The parallel performance of our algorithm also depends on the scale of flow fields. Fig. 21 depicts four typical curves of speedup, which are obtained from flow fields that have the same structure but different scales. For Figs. 21a, 21b, 21c, and 21d, the field domains are $[0, 69] \times [0, 69]$, $[0, 690] \times [0, 690]$, $[0, 1380] \times [0, 1380]$, and $[0, 2760] \times [0, 2760]$, respectively. The structure of the flow fields can be seen from Fig. 14. All involved placements take the same parameter settings without topological initialization. The numbers of streamlines placed for each field are

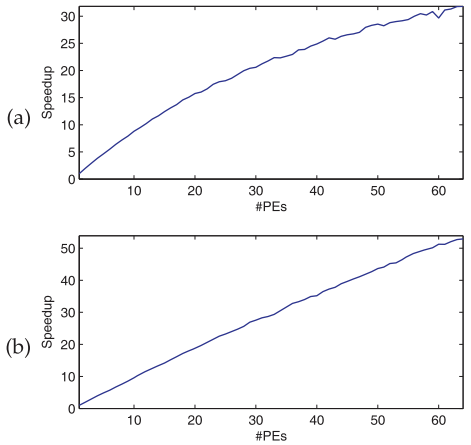


Fig. 20. Curves of speedup obtained from an enlarged version of the field in Fig. 14. The field domain is $[0, 2760] \times [0, 2760]$, and the placement settings are: $\gamma = 1$, and $\lambda = 0.2$. (a) is the original speedup, which is later improved to (b) by topological initialization.

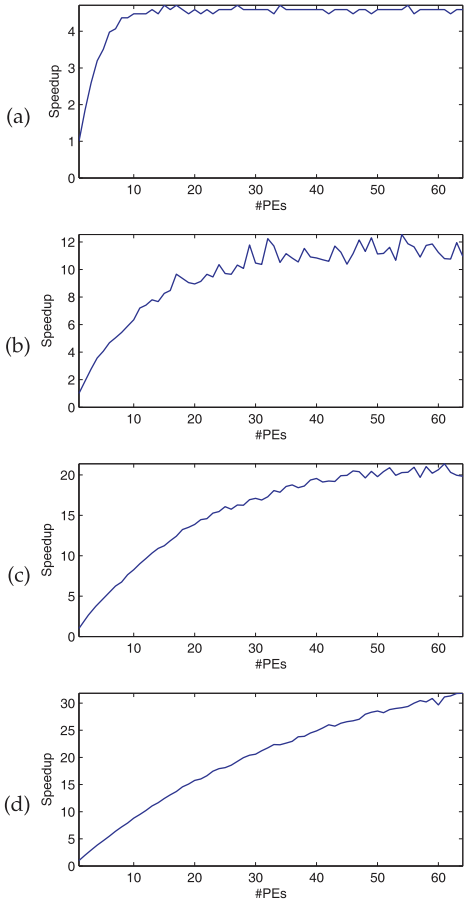


Fig. 21. Curves of speedup obtained from flow fields that have the same structure but different scales. The field domains are: (a) $[0, 69] \times [0, 69]$, (b) $[0, 690] \times [0, 690]$, (c) $[0, 1380] \times [0, 1380]$, and (d) $[0, 2760] \times [0, 2760]$. All involved placements take the same parameters, and no topological initialization is applied.

1. 314,
2. 3,916,
3. 7,961, and
4. 16,295.

The corresponding runtimes measured in ms with one PE are

1. 179,
2. 24,295,
3. 174,637, and
4. 1,187,514.

As shown in Fig. 21, the progressive improvement of speedup, which is achieved by simultaneously increasing the number of PEs and the size of flow fields, indicates that *our parallel algorithm is scalable*. Other similar tests also support this view.

6.2.3 Workload

Load balancing is an important issue of parallel computing. In our implementation of the algorithm, a task queue is used to schedule tasks among PEs in a granularity of LTAs. Each task created with an LTA only places one streamline. The total tasks are equal to the number of streamlines. According to the task scheduling, workload will be automatically balanced among PEs. For instance, we

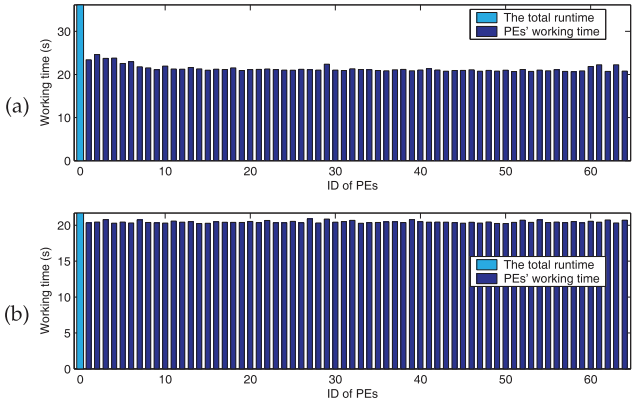


Fig. 22. The workload of all PEs that have worked for a test field. Both (a) and (b) have the same placement settings, but (a) is for a general case without topological initialization, while (b) is for a case approximating to the best where all PEs are nearly in full load because of topological initialization.

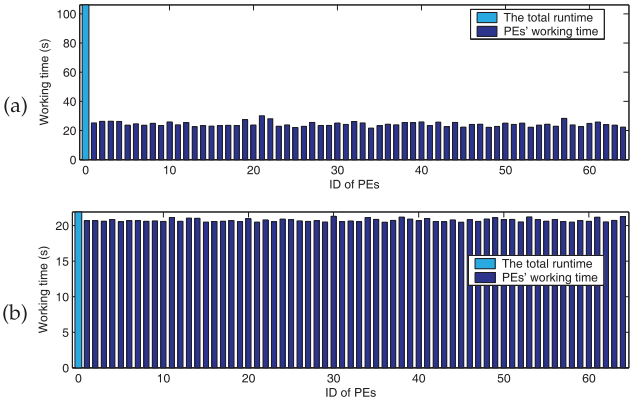


Fig. 23. The workload of all PEs that take a random seeding strategy to place streamlines for the same field as used in Fig. 22. The case without topological initialization is show in (a), while (b) is for the case with topological initialization.

measured the working time of 64 PEs that had worked for the speedup in Fig. 20a, and rendered the resulting histogram in Fig. 22a, where we can see that all PEs' working time is about the same, indicating that the workload is balanced approximately. In Fig. 22a, however, none of PEs is in full load, which degrades the parallel performance.

As discussed in Section 6.2.1, the parallel performance depends on the hierarchical decomposition of flow fields. The ideal case is that each streamline partitions its affiliated LTA into two or more equal parts. In this case, the number of LTAs will increase exponentially, and *all PEs will be in full load all the time* except for the stages of start-up and termination. The worst case is that there is only one LTA available all the time and hence *there is only one PE busy*. However, the worst case is very rare in practice, meanwhile the ideal case is hard to achieve, because LTAs may be in any shape and size. When the structure of the flow field is obtained by the aforementioned topological analysis, our algorithm based on the heuristic seeding strategy can achieve a parallel performance approximating to the best. For example, when topological initialization is applied, the histogram of working time in Fig. 22a is changed into that in Fig. 22b, where all PEs are nearly in full load. This result close to the best performance not only validates our heuristic seeding strategy, but also demonstrates that the parallel performance is mainly determined by the structure of flow fields although the seeding strategy has effect on that.

7 Discussions

To the best of our knowledge, our work is the first attempt on the parallelism of streamline placement. The proposed parallel algorithm can effectively speed up the placement of streamlines without introducing artificial boundaries and visual clutters caused by improper parallelization. The underlying parallel strategy is straightforward in theory. The key of its current implementation is the cell-based mechanism that is used to model isolation zones, saturation zones, LTAs, and VSAs. Although the cell-based models are not very precise, they are technically feasible just like the rasterization of graphic objects. With the cell-based models, we are in fact placing streamlines within different cell-based LTAs simultaneously and independently, where no synchronization or communication is required. The resulting placements are the same as those obtained by the sequential version of the algorithm.

In addition, the cell-based models are used to control the density of streamlines and accelerate placements at the same time. This technique is similar to the threshold distance control in [4], where grid cells are used to govern the tracing of streamlines. The difference is that our method improves the efficiency by replacing the distance computation with a cell-marking technique. When parameters λ and γ are specified, the minimum interstreamline distance $d_{\min}$ will not be less than λ except in neighborhood of singularities, while the maximum interstreamline distance $d_{\max}$ will not be more than $2\sqrt{2}\gamma$. By adjusting λ and γ, we can obtain placements in different styles as shown in Figs. 13 and 14. In order to maintain the embedded relationship of LTAs and VSAs, we must keep the constraint of $\gamma = n * \lambda$, where n is an integer not less than three.

Our parallel processing is based on an iterative decomposition of flow fields which is driven by streamlines. It can run without any information about topology though our idea is inspired by topological analysis. If the topological structure is available as an initial partitioning of the flow field, it will run better for its highly parallel performance as demonstrated in Figs. 19 and 22. When the field structure is unknown, the parallel performance may be low for some complex flow fields. The main reason is that the initially placed streamlines can not effectively partition the flow field into proper subdomains, which consequently limits the potential of parallel processing. The field partitioning especially in the early stage of the placement is mainly determined by the structure of the flow field, although the seeding strategy has effect on that. As evidence, we test a random seeding strategy that randomly selects seed points from LTAs, using the same test field and placement settings as those used for Fig. 22. The resulting histograms of PEs' working time are given in Fig. 23. Comparing Figs. 22 and 23, we can see that, the difference between the random seeding and our heuristic seeding is nearly negligible when topological initialization is applied to obtain the structure of

the flow field. When the structure is unknown, however, our seeding strategy is obviously better than the random seeding. This comparison also shows the feasibility of our heuristic seeding strategy. Except for the topological initialization, other techniques should be developed to improve initial partitioning of flow fields in future work.

As mentioned in Section 5, our parallel algorithm prefers shared memory systems, especially for multicore or many-core systems. For distributed memory systems, it is not very suitable due to the communication overhead caused by the dynamic partitioning of flow fields.

Currently, the algorithm is limited to 2D flow fields, because the domain decomposition is driven by streamlines. For 3D flow fields, it is unlikely to partition a 3D LTA into sub-LTAs by a single streamline. To partition 3D flow fields, stream surfaces may be a possible solution worthy of further study, on the basis of which, we can extend the basic ideas of isolation zones, saturation zones, LTAs, and VSAs to 3D flow fields. By generating stream surfaces along 3D topological skeleton, a 3D flow field could be partitioned into subregions that can be processed in parallel. However, topological analysis of 3D flow fields is challenging. Even if topology is available, there are still other obstacles such as rapidly increased memory requirement.

8 Conclusions

We developed a parallel streamline placement method for 2D flow fields. The parallel strategy is based on the proposed concept of local tracing areas, whose local tracing property of streamlines enables us to independently place streamlines within different LTAs and iteratively partition flow fields into hierarchical LTAs. In order to make streamlines uniformly spaced as possible, we also introduced the concepts of isolation zones, saturation zones, and valid seeding areas. All of them, including LTAs, are implemented as cell-based models established on orthogonal grids (i.e., the tracing grid and the seeding grid). With the help of the cell-based models, we proposed a heuristic centroid-based strategy to seed streamlines within LTAs, and developed a cell-marking technique to control the seeding and tracing of streamlines. Finally, a full algorithm for parallel streamline placement was presented in this paper.

We tested the algorithm on shared memory systems. Test results show that the algorithm can speed up the placement of streamlines effectively and overcome artificial boundaries and visual clutters caused by improper parallelization. The quality of placements is comparable to that obtained by existing placement methods. Currently, there are still some limitations to be addressed in possible future work.

Acknowledgments

The authors would like to thank the anonymous reviewers for their comments and suggestions. They would also like to thank G. Turk et al. and A. Mebarki et al. for providing their open source codes and test data sets. This work was supported in part by the National Natural Science Foundation of China under contracts 10972041 and 61272359.

References

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

[2] 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.

[3] B. Jobard and W. Lefer, "Creating Evenly-Spaced Streamlines of Arbitrary Density," *Proc. Eighth Eurographics Workshop Visualization in Scientific Computing,* vol. 7, pp. 43-56, 1997.

[4] Z. Liu, R.J. Moorhead II, and J. Groner, "An Advanced Evenly-Spaced Streamline Placement Algorithm," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 965-973, Sept./Oct. 2006.

[5] B. Spencer, R.S. Laramee, G. Chen, and E. Zhang, "Evenly Spaced Streamlines for Surfaces: An Image-Based Approach," *Computer Graphics Forum,* vol. 28, no. 6, pp. 1618-1631, 2009.

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

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

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

[9] G. Chen, K. Mischaikow, R.S. Laramee, P. Pilarczyk, and E. Zhang, "Vector Field Editing and Periodic Orbit Extraction Using Morse Decomposition," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 4, pp. 769-785, July/Aug. 2007.

[10] L. Li, H.-H. Hsieh, and H.-W. Shen, "Illustrative Streamline Placement and Visualization," *Proc. IEEE Pacific Visualization Symp.,* pp. 79-86, 2008.

[11] Y. Chen, J.D. Cohen, and J.H. Krolik, "Similarity-Guided Streamline Placement with Error Evaluation," *IEEE Trans. Visualization and Computer Graphics,* vol. 13, no. 6, pp. 1448-1155, Nov./Dec. 2007.

[12] W. Zhang and J. Deng, "Topology-Driven Streamline Seeding for 2D Vector Field Visualization," *Proc. IEEE Int'l Conf. Systems, Man and Cybernetics,* pp. 4901-4905, 2009.

[13] K. Wu, Z. Liu, S. Zhang, and R.J. Moorhead II, "Topology-Aware Evenly Spaced Streamline Placement," *IEEE Trans. Visualization and Computer Graphics,* vol. 16, no. 5, pp. 791-801, Sept./Oct. 2010.

[14] W. Zhang, B. Sun, and Y. Wang, "A Streamline Placement Method Highlighting Flow Field Topology," *Proc. Int'l Conf. Computational Intelligence and Security,* pp. 238-242, 2010.

[15] H. Yu, C. Wang, C. Shene, and J. Chen, "Hierarchical Streamline Bundles," *IEEE Trans. Visualization and Computer Graphics,* vol. 18, no. 8, pp. 1353-1367, Aug. 2012.

[16] L. Xu, T.-Y. Lee, and H.-W. Shen, "An Information-Theoretic Framework for Flow Visualization," *IEEE Trans. Visualization and Computer Graphics,* vol. 16, no. 6, pp. 1216-1224, Nov./Dec. 2010.

[17] K.L. Ma, J. Painter, C.D. Hansen, and M.F. Krogh, "Parallel Volume Rendering Using Binary-Swap Compositing," *IEEE Computer Graphics and Applications,* vol. 14, no. 4, pp. 59-67, July 1994.

[18] B. Cabral and C. Leedom, "Highly Parallel Vector Visualization Using Line Integral Convolution," *Proc. Seventh SIAM Conf. Parallel Processing for Scientific Computing,* pp. 803-807, 1995.

[19] M. Zöckler, D. Stalling, and H. Hege, "Parallel Line Integral Convolution," *Parallel Computing,* vol. 23, no. 7, pp. 975-989, 1997.

[20] H. 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, Apr.-June 1998.

[21] S. Bachthaler, M. Strengert, D. Weiskopf, and T. Ertl, "Parallel Texture-Based Vector Field Visualization on Curved Surfaces Using GPU Cluster Computers," *Proc. Eurographics Symp. Parallel Graphics and Visualization,* pp. 75-82, 2006.

[22] D. Sujudi and R. Haimes, "Integration of Particle and Stream Lines in a Spatially-Decomposed Computation," *Proc. Parallel Computational Fluid Dynamics,* 1996.

[23] L. Chen and I. Fujishiro, "Optimizing Parallel Performance of Streamline Visualization for Large Distributed Flow Datasets," *Proc. IEEE VGTC Pacific Visualization Symp. '08,* pp. 87-94, 2008.

[24] H. Yu, C. Wang, and K.L. Ma, "Parallel Hierarchical Visualization of Large Time-Varying 3D Vector Fields," *Proc. ACM/IEEE Conf. Supercomputing '07,* pp. 1-12, 2007.

[25] D. Pugmire, H. Childs, C. Garth, S. Ahern, and G. Weber, "Scalable Computation of Streamlines on Very Large Datasets," *Proc. ACM/IEEE Conf. Supercomputing '09*, pp. 16:1-16:12, 2009.
[26] T. Peterka, R. Ross, B. Nouanesengsey, T.-Y. Lee, H.-W. Shen, W. Kendall, and J. Huang, "A Study of Parallel Particle Tracing for Steady-State and Time-Varying Flow Fields," *Proc. 21st Int'l Parallel and Distributed Processing Symp. (IPDPS '11)*, pp. 580-591, May 2011.
[27] J.C. Butcher, *Numerical Methods for Ordinary Ddifferential Equations*, pp. 59-95. Wiley, 2003.
[28] J. Helman and L. Hesselink, "Representation and Display of Vector Field Topology in Fluid Flow Data Sets," *Computer*, vol. 22, no. 8, pp. 27-36, Aug. 1989.
[29] D. Stalling and H. Hege, "Fast and Resolution Independent Line Integral Convolution," *Proc. ACM SIGGRAPH '95*, pp. 249-256, 1995.
[30] G. Turk and D. Banks, "Streamline Package," http://www.cc.gatech.edu/~turk/streamlines/streamlines.html, 2011.
[31] A. Mebarki, "Streamline Package," http://amebarki.visiondz.info/Ressources/, 2011.
[32] G. Scheuermann, H. Hagen, and H. Krüger, "An Interesting Class of Polynomial Vector Fields," *Mathematical Methods for Curves and Surfaces II*, M. Dæhlen, T. Lyche, and L. L. Schumaker, eds., pp. 429-436, Vanderbilt Univ. Press, 1998.
[33] A. Mebarki, "2D Placement of Streamlines," http://www.cgal.org/Manual/latest/doc_html/cgal_manual/Stream_lines_2/Chapter_main.html, 2011.
[34] B. Nouanesengsy, T.-Y. Lee, and H.-W. Shen, "Load-Balanced Parallel Streamline Generation on Large Scale Vector Fields," *IEEE Trans. Visualization and Computer Graphics*, vol. 17, no. 12, pp. 1785-1794, Dec. 2011.
[35] M. Hlawatsch, F. Sadlo, and D. Weiskopf, "Hierarchical Line Integration," *IEEE Trans. Visualization and Computer Graphics*, vol. 17, no. 8, pp. 1148-1163, Aug. 2011.

Wenyao Zhang received the MS degree in pattern recognition and intelligent system from Beijing Institute of Technology, China, in 1999, and the PhD degree in computer science and technology from Chinese Academy of Sciences, China, in 2003. He is currently an assistant professor with the School of Computer Science, Beijing Institute of Technology. He is also a member of Beijing Laboratory of Intelligent Information Technology. His current research interests include scientific visualization, media computing, and parallel processing.

Yi Wang received the BS and the MS degrees in computer technology and application from Air Force Engineering University of China in 1996 and 2004, respectively. He is currently a doctoral candidate at Beijing Institute of Technology, with a dissertation on the topic of scientific visualization and its applications in computational mechanics.

Jianfeng Zhan received the PhD degree in computer software and theory from the Chinese Academy of Sciences, Beijing, China, in 2002. He is currently a professor of computer science with the Institute of Computing Technology, Chinese Academy of Sciences. His current research interests include distributed and parallel systems, operating systems, and computer architecture. He is one of the core members of the petaflops HPC system and data center computing projects at Institute of Computing Technology, Chinese Academy of Sciences. He was a recipient of the Second-class Chinese National Technology Promotion Prize in 2006, and the Distinguished Achievement Award of the Chinese Academy of Sciences in 2005.

Beichen Liu received the BS degrees in digital media technology from Beijing University of Technology, Beijing, China, in 2011. He is currently a graduate with the School of Computer Science, Beijing Institute of Technology. His research interests include flow visualization and image processing.

Jianguo Ning received the BS degree in mechanics and the MS degree in solid mechanics from Lanzhou University, China, in 1985 and 1988, respectively, and the PhD degree in solid mechanics from Taiyuan University of Technology, China, in 1992. He is currently a professor with Beijing Institute of Technology. His current research interests include explosion mechanics, impact dynamics, computational mechanics, and scientific visualization. He has published more than 150 peer-reviewed papers. He was supported by National Outstanding Youth Fund and Chang Jiang Scholar program in 2006 and 2007, respectively. He is the vice-director of Applied Mechanics Professional Committee of China Ordnance Society, member of Calculating Mechanics Professional Committee. He is a member of the editorial board of the *Journal of Computational Mechanics*, and *Explosion and Shock Waves*.