Hierarchical LIC for Vector Field Visualization

Udeepta Bordoloi and Han-Wei Shen
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210
E-mail: bordoloi@cis.ohio-state.edu and hwshen@cis.ohio-state.edu

Abstract

This paper presents a hierarchical algorithm to accelerate 2D LIC computation. A quadtree data structure, combined with vector field simplification metrics, are employed to provide the capability of selective LIC approximation. In the algorithm, each node of the quadtree is associated with a measure of "complexity" corresponding to the local flow field. At run time, a threshold is provided by the user to determine the degree of approximation. We report work in progress aiming to solve two fundamental problems: (1) Find an appropriate metric as a measure of the degree of vector field complexity. (2) Develop an approximate LIC algorithm to produce an image that gives a faithful representation of the vector field, i.e., it should have as much information as a normal LIC image.

1 Introduction

One of the most popular methods for vector field visualization is Line Integral Convolution, or LIC [1]. Since this method was first introduced in 1993, researchers have proposed many extensions to improve the computation speed [2, 3], to produce better LIC images [4, 5, 6], and to apply LIC to both steady and unsteady flow fields [7, 8, 9]. The popularity of LIC mainly comes from its effectiveness in depicting the flow directions everywhere in a dense vector field. The disadvantage of LIC, however, is that it is computationally expensive. Even though computing power has increased significantly in the last decade, the amount of data generated from numerical simulations has also increased by many orders of magnitude. In this paper, we present an algorithm that utilizes a hierarchical scheme to accelerate LIC computation, and to make interesting features of a vector field stand out against the relatively uninteresting background.

The main cost of computing LIC images is streamline advection. To reduce the computation time, previously Stalling and Hege have proposed a fast-LIC method [2] which performs convolution incrementally for pixels along a streamline to reduce the overall number of streamlines being computed. In addition, adaptive step size control for a Runge-Kutta numerical integration method is also employed by the fast-LIC method to further speed up the computation. In contrast with the fast-LIC method, the technique presented in this paper adopts a different approach. Instead of computing accurate streamlines everywhere to cover the entire field, streamlines are computed approximately and selectively based on the underlying vector field features. In parts of the vector field where the flow directions are fairly straight or similar across a local region, only one streamline is computed and the same streamline is used by the points in the entire neighborhood to perform convolution. Our goal is to reduce the number of streamlines computed and simplify the LIC computation, and thus reduce the total computation cost.

To make the algorithm suitable for applications that have different visual quality and interactivity requirements, it is very important that different levels of approximations to be provided and controlled. To achieve this goal, we use a *quadtree* hierarchical data structure to provide the capability of selective approximation when computing 2D LIC images. In our algorithm, each node of the quadtree is associated with a measure of "complexity" corresponding to the local flow field. At run time, the user provides a threshold to specify the minimum complexity level to displayed. If the measure associated with a given node is lower than the user supplied value, the corresponding flow field region is simplified and approximate LIC is performed in the region. Otherwise, the algorithm subdivides the region into smaller parts and performs the same test recursively. To make possible such a hierarchical LIC computation, two key issues need to be addressed:

(1) Find an appropriate metric as a measure of the degree of complexity for the local areas of the underlying vector field.

(2) Develop an approximate LIC algorithm to produce an image that gives a faithful representation of the vector field, i.e., it should have as much information as a normal LIC image.

In the following, we present our hierarchical LIC algorithm addressing the above two problems. We first briefly overview the LIC algorithms. We then describe the error metrics that are used in the quadtree hierarchical data structure. Two approximate methods for LIC computation are discussed, and experimental results demonstrating both the image quality and computational speed of the algorithm are presented.

1.1 Related Work

The Line Integral Convolution method is a texture synthesis technique that can be used to visualize two-dimensional vector field data. Taking as the input a vector field and a white noise image with the same resolution as the vector field, LIC computes convolution of the input noise image using the following algorithm: For each pixel, streamlines in both the positive and negative directions are first calculated. The pixel's convolution result is computed by weighted-averaging the image values of the pixels along the streamline paths. As a result, the intensity values of the pixels along each streamline are strongly correlated so the directions of the flow field can be clearly visualized.

While LIC is effective in visualizing 2D vector fields, it is quite computationally expensive. Stalling and Hege proposed an extension to speed up the process [2]. Their work is based on two key observations. First, a streamline starting from any point in the domain actually passes through many pixels. For those pixels, only one rasterized streamline is sufficient to produce the convolution and thus redundant numerical integrations can be avoided. The second observation is that adjacent pixels along the same streamline use very similar sets of pixel values for the convolution. Therefore, the LIC value computed for one pixel can be reused by its neighbors with small modifications to accelerate the convolutions. By reducing the number of streamlines computed and speeding up the LIC convolution, Stalling and Hege's new method can gain a great saving in computing the LIC.

The main idea of this paper is to further speed up the LIC computation by adopting vector field simplification methods and hierarchical data structures [10, 11]. While the remaining of the paper discusses our algorithm for accelerating the standard LIC method, we believe that combining our algorithm with the fast-LIC algorithm can be very effective.

1.2 Measure for simplification

To allow different levels of approximations, we need to provide error measures to characterize the degree of *complexity* in the regions of the vector field. We use the term "complexity" to measure the parallelism between the streamlines, i.e., how the flow directions in a local region are similar to each other. We have implemented two measures to represent the complexity of the vector field. One is the magnitude of *curl*, and the other is the metric proposed by Heckel *et al.*[10]. Our goal is to make regions with features like vortices and saddle points have a high degree of complexity. Uninteresting parts such as straight flows, on the other hand, are considered to have a low degree of complexity.

The measures are calculated for each point in the field at a preprocessing stage, and a quadtree is built out of the information. During the actual LIC image computation stage, we traverse the quadtree in a depth first manner. We adopt the convention that a low value of the measure implies that the region can be simplified, and a high value suggests that there might be interesting features in the region, and we should subdivide the region further, i.e., go down the current quadtree level. Thus the measure we are using denotes 'complexity'. We do not simplify a region even if only one point within it has a 'complexity' value higher than the user defined threshold. This is achieved by keeping track of the maximum value of 'complexity' for each region, and it is this value that is kept at each node of the quadtree.

1.2.1 Curl

The curl at a point in a vector field is defined as the cross product between the divergence of the vector field at that point and the vector given by the field at that point. Mathematically, it has the form:

$$Curl(x, y) = (\frac{dF_y}{dx} - \frac{dF_x}{dy})\mathbf{z}$$

where F_x and F_y are the x and y components of the vector. A intuitive geometrical interpretation of curl at a point would be that it gives a measure of how much the vector field curls around that point. Thus the magnitude of curl at the center of a whirlpool would be very high, and that of a point in the middle of a straight flow would be zero. The quadtree preprocessing for curl is straightforward. We first calculate the magnitude of curl for each point in the vector field, and then build the quadtree over the vector field in a bottom up fashion. Each node of the tree stores the maximum value of curl in the region the node represents.

The magnitude of curl represents a local measure of complexity for the flow, as only the vectors from a point's adjacent points are used in the calculation. A very low curl at a point means that the flow is almost parallel at that point. But it would not necessarily mean that the flow is parallel some distance from the point. Potentially, this nature of locality can become problematic when simplifying the LIC computation as the streamline convolution path originating from each point has a much wider span. Some experimental results are presented and discussed in the result section.

1.2.2 Streamline Distance Error

Another flow complexity measure that we chose to use was proposed by Heckle *et al.*[10]. For each point, the metric represents the

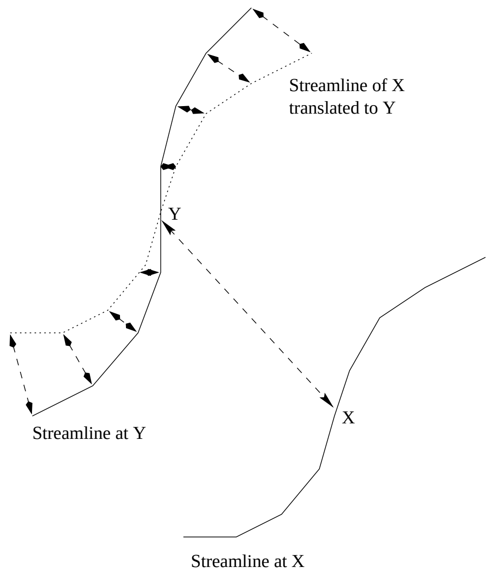

Figure 1: Streamline distance error metric

difference between the accurate and the approximate streamlines originating form the point in question. More specifically, the error is measured as the sum of distances of the corresponding points on the two streamlines, as shown in Figure 1.

Generating a quadtree out of this measure is slightly different from the maximum quadtree method that we use for the curl metric. At the lowest level of the quadtree, say a 2×2 block, we calculate the error at each point as the distance between the actual streamline originating from that point and the approximate streamline, which is a translated streamline computed form the center of the 2×2 block. The leaves store the maximum of the error distances of the four points in the 2x2 block. For the next level, we follow the same steps for the 4x4 region a node represents. This time, the value at the node is the maximum of the error distances of the sixteen points in the region, and the error distances are computed with respect to the streamline of the center point of this region. Note that this center point is different from any of the center points of the 2x2 sub-regions of this region.

To approximate a LIC image, this metric is believed to be more intuitive if we are to translate an approximate streamline across a region to perform the convolution. If the translated streamline does not model the actual streamline well, the distance would be high and thus we need to use a finer level of quadtree approximation. Before we present the approximation result using this metric, we first we discuss our LIC approximation methods in the next section.

1.3 Approximating LIC

For a region that has a low complexity measure, we intend to limit the calculation of streamlines as much as possible, i.e., use only one (or possibly a few) streamline to approximate the flow directions for the whole region. In the following we describe two methods for approximating LIC using the quadtree-based hierarchical vector field simplification measure that we have discussed.

1.3.1 Streamline Translation

Our first method is to translate an approximate streamline to each point of a local region and use the approximate streamline trajectory as the LIC convolution path. For a region that is determined to

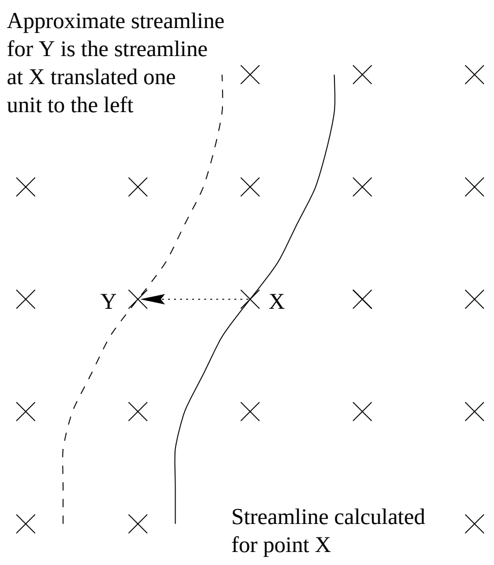

Figure 2: LIC approximation using streamline translation

be simplified, we select one point, near the center of the region, and calculate the actual streamline originating from this point and perform rasterization of the streamline. To calculate LIC for any other point in the region, we translate the rasterized streamline from the center point by a displacement vector (the vector from the central point to the current point that we are approximating the streamline for), and use it to perform the same convolution as the standard LIC.

Figure 2 shows an example of our algorithm. Consider the grid point that is to the left of the current point. Since we assume that the shape of the streamlines is similar, we just translate the current streamline one unit to the left to get the approximate streamline for the new point.

1.3.2 Pseudo LIC

We have also experimented with Pseudo LIC (PLIC) proposed by Verma *et al.*[12], which uses texture mapping to generate LIC-like textures. PLIC places seed points for streamlines on a uniform grid, which might be sparser than the grid of the output image. (Compare this to the traditional method of LIC, in which seed points for streamlines are pixels of the output image). To generate flow textures, a rectangular LIC texture of a straight field is warped and texture mapped onto the streamline. The quality of PLIC output depends on how sparse the seed points are, and how wide and how long the rectangular patch used for texture map is. More details can be found in [12].

In our algorithm, the streamline is calculated at the central point of the region to be simplified. Instead of performing regular LIC convolution for each point in the region, we texture map a rectangular patch from a previously calculated LIC image of a straight vector field onto the streamline. The catch is that the seed points are no longer uniformly spaced, and we need to calculate a proper width and length for the rectangular patch depending on the size of the region we are simplifying.

2 Results and Discussion

In the following, we first give qualitative assessments on the merits and demerits of the metrics we use, and then discuss the speedups and the result images.

We have applied our algorithm to a 400×400 two dimensional vector field with three vortices and two saddle points. Figures 3-6 show results using the streamline translation method for LIC approximation with the curl and the streamline distance as the error metrics. Figures 7-10 were generated using PLIC approximation with the same set of error metrics. Figure 11 is the traditional LIC image for comparison. The streamline distance error metric is calculated using a streamline advected to a length of 40 units. The images generated using the streamline translation method use a convolution length of 40 units, while the Pseudo LIC images are convolved to 20 units.

From the figures we note that when the curl is used as a metric, the image begins to show noticeable artifacts around the vortices. The reason for this can be better understood from the image of the magnitude of the curl of the vector field in Figure 12. The curl is very high at the saddle points and also at the centers of the vortices. But as we move away from the vortex centers, the curl falls to levels that are comparable to areas with a straight flow. This happens because curl is calculated locally, while the streamline calculated for generating the LIC images spans a much greater region. Thus one of the drawbacks of curl is that it fails to capture the global features to the field.

The streamline distance error metric is very well suited for the translation method of approximating streamlines. Since the error is calculated using the same translated streamline that is later used for approximation, it tells us exactly how good or bad the approximation is. The effects of simplification when using the streamline distance error metric, however, start to appear near the saddle points. From the images of the various levels of the distance error in Fig 13 and 14, we notice that the error is high for the vortices but really low around the saddle points.

Based on the above observations, neither curl nor the streamline distance error appear to be the perfect metric for vector field simplification on their own right. It is likely that a combination of both the curl and the streamline distance would prove to be much more faithful in representing the ’complexity’ of regions of vector fields.

Using either of the LIC approximation methods, appreciable speedup can be achieved (see Figure captions) with different degrees of convolution artifacts. The tradeoff between image quality and computation speed is controlled by the user. The streamline translation method produces artifacts as the coherence becomes weaker between the pixels that are adjacent to each other but in different regions of the quadtree simplification. In addition, the correlation between the pixels in the region that is simplified is also disturbed as the convolution paths for the pixels that are along the same streamline in the original field now become slightly different as a result of the streamline approximation. However, for the flow regions that have low complexity, the differences are small in general. Pseudo LIC seems to be a much better candidate for approximation. We do not see discontinuity along the boundaries of the regions that are simplified. In the original Pseudo LIC algorithm, streamlines are uniformly spaced. For our purposes, we have to place them according to the quadtree traversal, which results in non-uniform placement. This causes different parts of the final image to have different intensities because they have different amounts of texture mapped values deposited on them. We solve this problem by starting new streamlines from points which do not have a minimum number of texture mapped deposits on them.

3 Future Work

There is scope for more work on both the error metrics and the LIC approximation methods. We believe we can produce a better metric by combining more than one property of the vector field, in this case the curl and the streamline distance. As far as approximation is concerned, we can use concepts of fast-LIC in our algorithm to achieve higher speedups. Finally, the use of a hierarchical algorithm should prove very useful for producing three dimensional LIC images, where the challenge is both computational speed and selective display of information.

Acknowledgments

This work was supported by The Ohio State University Research Foundation Seed Grant. We would like to thank Ravi Samtaney for providing the test data sets and Dr. Roger Crawfis for useful comments.

References

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

[2] D. Stalling and H.-C. Hege. Fast and resolution independent line integral convolution. In *Proceedings of SIGGRAPH 95*, pages 249–256. ACM SIGGRAPH, 1995.

[3] M. Zöckler, D. Stalling, and H.-C. Hege. Parallel line integral convolution. In *Proceedings of First Eurographics Workshop on Parallel Graphics and Visualisation*, pages 111–128, September 1996.

[4] M.-H. Kiu and D. Banks. Multi-frequency noise for LIC. In *Proceedings of Visualization '96*, pages 121–126. IEEE Computer Society Press, Los Alamitos, CA, 1996.

[5] H.-W. Shen, C.R. Johnson, and K.-L. Ma. Visualizing vector fields using line integral convolution and dye advection. In *Proceedings of 1996 Symposium on Volume Visualization*, pages 63–70. IEEE Computer Society Press, Los Alamitos, CA, 1996.

[6] V. Interrante and C. Grosch. Strategies for effectively visualizing 3d flow with volume lic. In *Proceedings of Visualization '97*, pages 421–424. IEEE Computer Society Press, Los Alamitos, CA, 1997.

[7] L.K. Forssell and S.D. Cohen. Using line integral convolution for flow visualization: Curvilinear grids, variable-speed animation, and unsteady flows. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):133–141, 1995.

[8] A. Okada and D. L. Kao. Enhanced line integral convolution with flow feature detection. In *Proceedings of IS&T/SPIE Electronic Imaging '97*, pages 206–217, 1997.

[9] H.-W. Shen and D.L Kao. A new line integral convolution algorithm for visualizing time-varying flow fields. *IEEE Transactions on Visualization and Computer Graphics*, 4(2), 1998.

[10] B. Heckel, G. Weber, B Hamann, and K. Joy. Construction of vector field hierarchies. In *Proceedings of Visualization '99*, pages 19–25. IEEE Computer Society Press, Los Alamitos, CA, 1999.

[11] A. Telea and J. van Wijk. Simplified representation of vector fields. In *Proceedings of Visualization '99*, pages 35–42. IEEE Computer Society Press, Los Alamitos, CA, 1999.

[12] V.. Verma, D. Kao, and A. Pang. Plic: Bridging the gap between streamlines and lic. In *Proceedings of Visualization '99*, pages 341–348. IEEE Computer Society Press, Los Alamitos, CA, 1999.

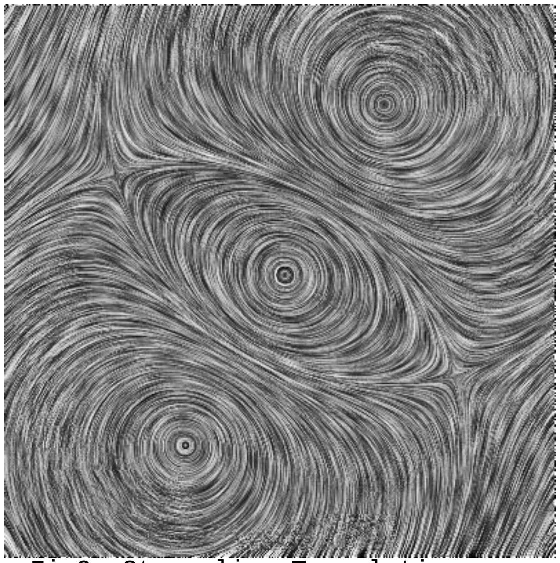
Fig3. Streamline Translation,
metric:curl, speedup:21.2%

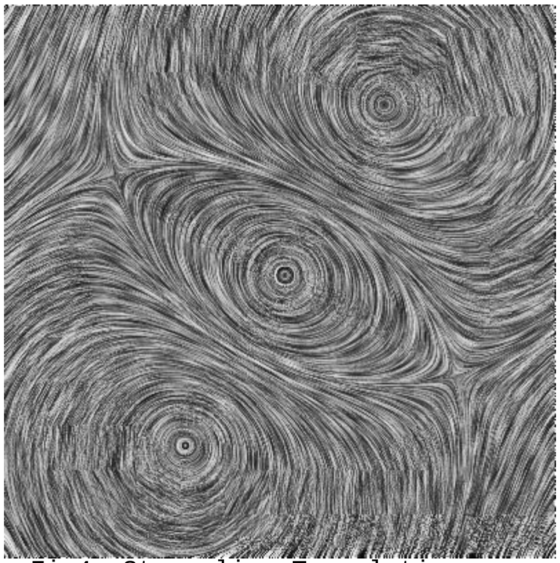
Fig4. Streamline Translation,
metric:curl, speedup:50.3%

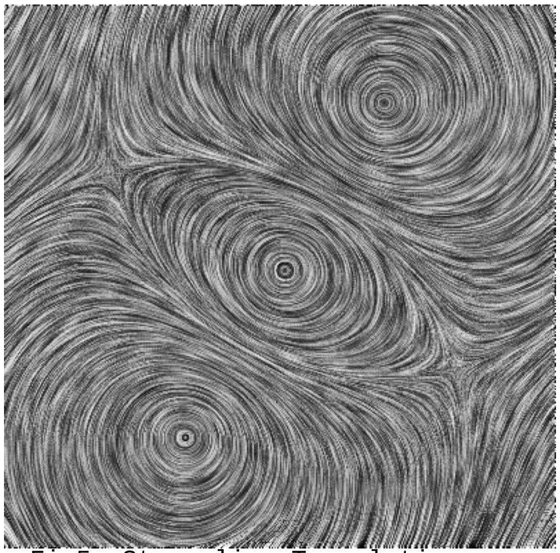
Fig5. Streamline Translation,
metric:distance, speedup:34.1%

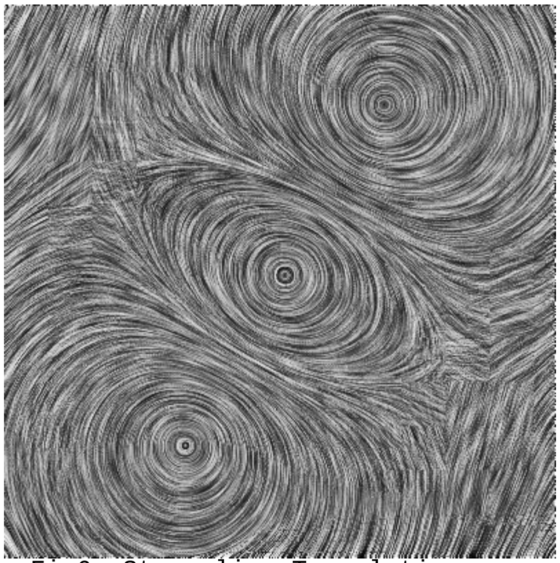
Fig6. Streamline Translation,
metric:distance, speedup:52.6%

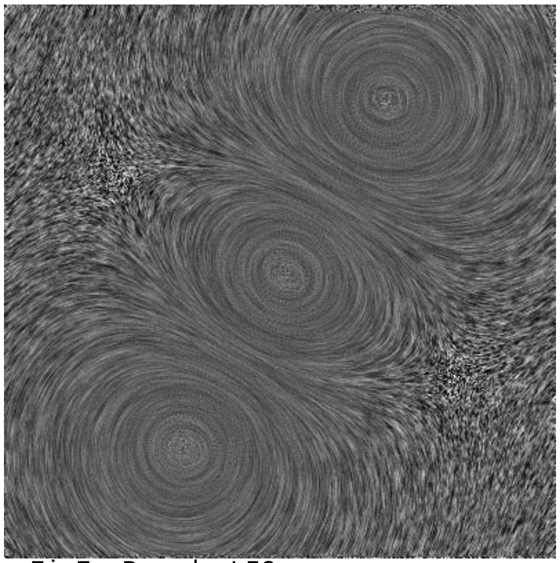
Fig7. Pseudo LIC,
metric:curl, speedup:23.6%

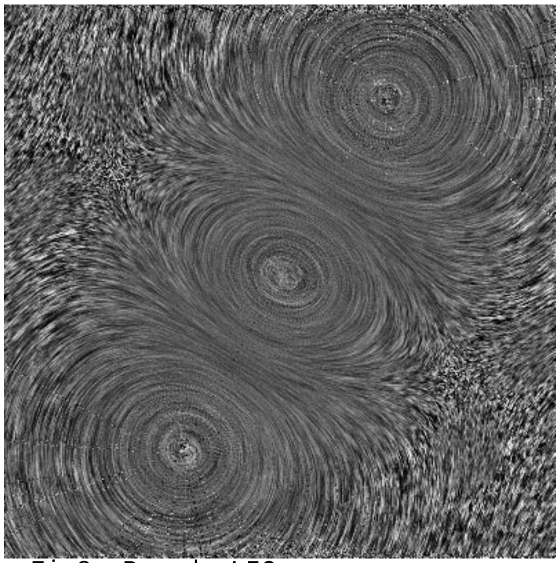
Fig8. Pseudo LIC,
metric:curl, speedup:43.3%

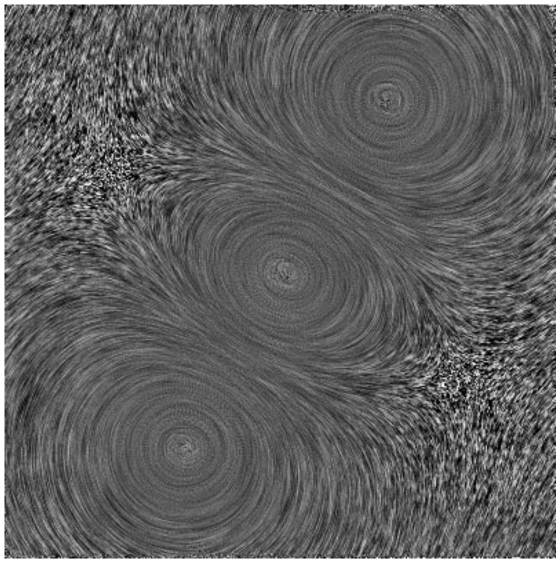
Fig9. Pseudo LIC,
metric:distance, speedup:25.3%

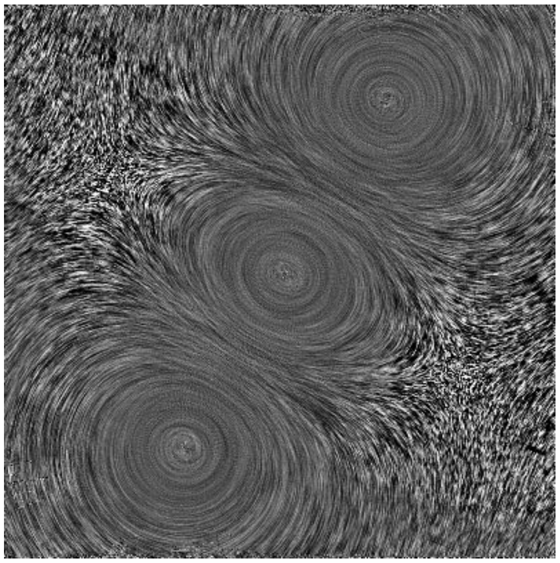
Fig10. Pseudo LIC,
metric:distance, speedup:47.5%

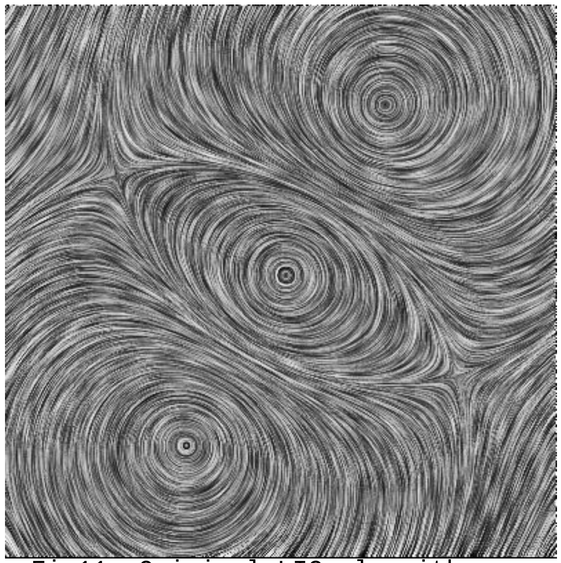
Fig11. Original LIC algorithm
Time: 33.6s

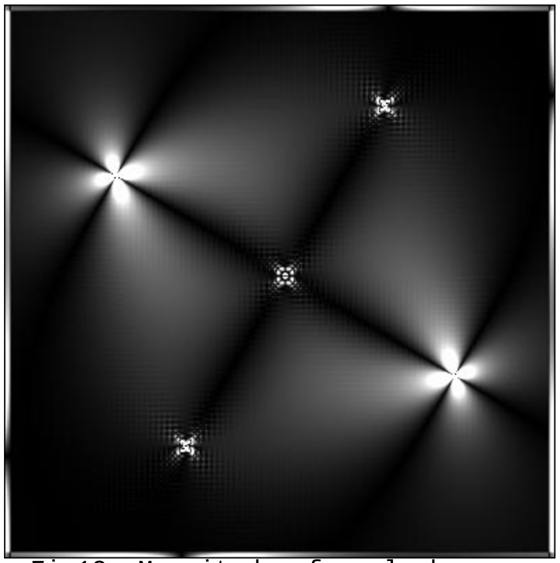
Fig12. Magnitude of curl shown
as a gray scale image

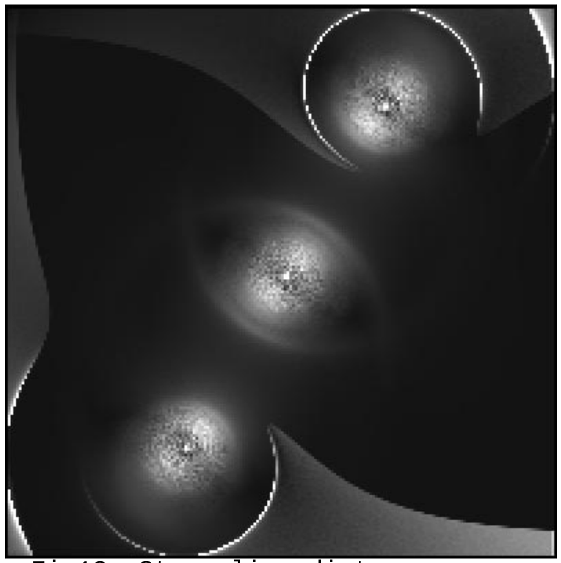
Fig13. Streamline distance error
for the level 2x2

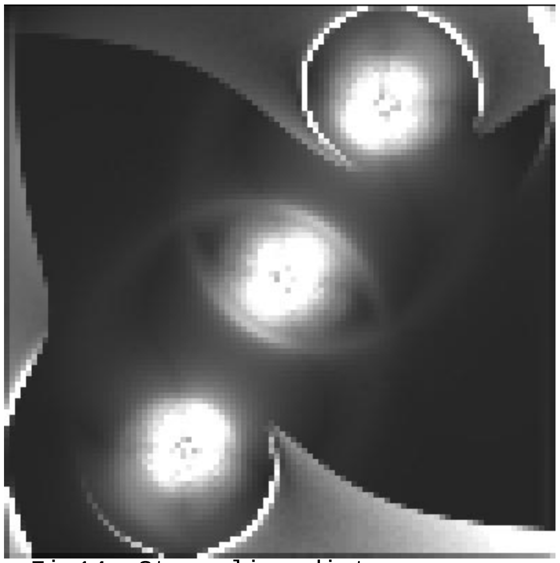
Fig14. Streamline distance error
for the level 4x4