Received January 7, 2019, accepted January 16, 2019, date of publication January 31, 2019, date of current version February 22, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2895857

A Line Integral Convolution Method With Dynamically Determining Step Size and Interpolation Mode for Vector Field Visualization

XUJIA QIN, XIA FANG, LOUHENG CHEN, HONGBO ZHENG, JI MA, AND MEIYU ZHANG
College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China

Corresponding author: Hongbo Zheng (zhb@zjut.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702455, Grant 61672462, and Grant 61672463.

ABSTRACT The line integral convolution method can obtain a continuous and dense vector streamline texture. The traditional fixed step size gets larger texture noise, while the exact calculation step method (such as the Runge–Kutta method) is time-consuming. In the process of generating streamline texture, if using bilinear interpolation calculation, we can get the method of calculating the point vector value, and it is also time-consuming. To address the above-mentioned problems, the line integral convolution method for dynamically determining integral step size and interpolation mode is proposed. The effect of streamline drawing mainly depends on the size of the integration step and the interpolation mode. The method proposed in this paper gives the rules for determining the integration step size and interpolation mode according to the vector speed size and the angle between two vectors. The experimental results show that compared with the previous fixed step method and the bilinear interpolation method, the proposed method not only has faster computation speed but also clearer texture and stronger contrast.

INDEX TERMS Vector field visualization, line integral convolution, streamline texture, integral step size.

I. INTRODUCTION

Vector fields are everywhere, and many natural phenomena can be described by vector fields. For example, the continuously dynamic processes of ocean currents and typhoon wind fields can be expressed by vector fields. With the help of computer graphics and image processing techniques, as well as the high-performance computing power of computers, these invisible vector fields can be simulated to form the research direction of vector field visualization.

The texture method expresses the vector field information in the form of texture, which has spatial continuity and can represent scalar information by color. Texture-based mapping methods mainly include: spot noise method [1] and line integral convolution (LIC) method [2]. This paper mainly introduces texture visualization of line integral convolution.

The spot noise method was proposed by van Wijk in 1991 and was the first method to use textures for visualization [1]. Spot noise is the convolution of points with certain size and shape along vector line, and the texture is controlled by changing the attributes of points. These points have the characteristics of random distribution. Carbral and Leedom [2] improved based on the spot noise method, and presented the line integral convolution method. The basic idea of the algorithm is to traverse the data sampling points in the vector field. From this point forward and backward, line integral convolution is carried out according to the corresponding vectors. The result is taken as the value of the point, and the convolution core uses one-dimensional kernel function. One main drawback of the LIC algorithm proposed by Carbral and Leedom is that it is inefficient. Stalling and Hege [3] proposed a fast convolution algorithm (Fast LIC) to improve the performance of the original LIC method. The original LIC method traverses each point to perform line integral convolution, in which there are a large number of repeated steps. By adjusting the order of calculation of the pixels, the amount of line integral calculation can be greatly reduced. The speed and accuracy of the LIC algorithm are improved by the fourth-order Runge-Kutta numerical integration method. Zöckler *et al.* [4] implemented parallel

The associate editor coordinating the review of this manuscript and approving it for publication was Mu-Yen Chen.

operations on distributed computers based on Fast LIC, which is called Parallel Fast LIC. Forssell [5] applied Fast LIC to curved meshes. Interrante and Grosch [6], [7] presented volume rendering method (Volume LIC) for three-dimensional data. Fast LIC was applied to three-dimensional volume data field to display three-dimensional data better through specific rendering strategies. Sundquist [8] proposed an extension of the fast convolution method (Dynamic LIC), which used two vector fields to represent dynamic electromagnetic fields. Rezk-Salama *et al.* [9] stated a texture-based volume rendering method that can interactively display the visualization of a three-dimensional vector field. Hlawatsch *et al.* [10] proposed a hierarchical line integration strategy to accelerate the integration process by using a multi-core architecture.

The image based flow visualization (IBFV) algorithm [6] implemented the flow field drawing by using the forward mesh projection mapping to realize the flow of noise textures and the generated frame image, and then combined the proportional mixture of the continuous frame images. Huang *et al.* [11] presented an output algorithm based on image spatial continuity to solve the visualization problem of surface flow field. Skraba *et al.* [12] offered a robust vector field visualization method designed to simplify vector field feature points. After that, the unsteady vector field simplification method is further identified [13]. The hierarchical line integration strategy proposed by Hlawatsch *et al.* [10] using a multi-core architecture to accelerate the integral calculation process in the LIC algorithm. Quan *et al.* [14] described a visualization method for field strength driving. Oeltze *et al.* [15] used a variety of clustering methods to cluster by defining similarity, and selected representative streamlines of each class for visualization.

In this paper, based on two-dimensional vector data, vector simulation and visualization based on texture are studied. Line integral convolution process in texture visualization is improved, and visualization effect and efficiency are improved.

II. LIC PRINCIPLE AND IMPLEMENTATION

A. LIC PRINCIPLE

Texture convolution is the calculation of texture convolution of an image based on vector fields and textures (white noise) to form a spatially continuous representation. The essence of line integral convolution is to generate streamlines and convolute the corresponding vectors on the streamlines to obtain the corresponding texture values of the image. The core of LIC is the generated streamline, and the key point for the effect and efficiency of the streamline is the integration step and sample point interpolation. According to the analysis of integration step and sampling point interpolation between two points, the generated textures with different integration step and sampling point interpolation strategies are directly given, and corresponding comparisons are made. The difference of texture and time between different methods can be seen in Figure 1.

The diagram of the LIC algorithm is shown in Figure 1.

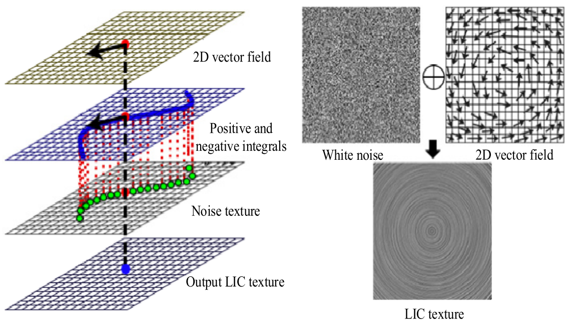


FIGURE 1. Diagram of LIC algorithm.

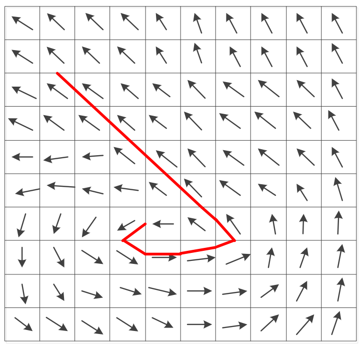


FIGURE 2. The calculation process of streamline in LIC algorithm.

B. LIC IMPLEMENTATION

In 1991, van Wijk [1] proposed the spot noise method, which is the first work for texture vector visualization. In 1993, Cabral and Leedom [2] first proposed the spot noise-based line integral convolution method. Its main idea is to generate a texture for the random white noise convolution along the streamlines of the varying vectors positive and negative directions. The resulting streamline of the texture is closely related to the changing vector directions.

The streamline generation and calculation process is shown in Figure 2.

The weight of each step of the texture convolution is as follows:

$$h_i = \int_{s_i}^{s_i+\Delta s_i} k(\omega)\, d\omega \tag{1}$$

In Eq. (1), Δs_i is the step length of step i in streamline convolution calculation, and s_i is the current streamline length of step i. The iterative computation method is shown in the following equation.

$$\begin{cases} s_0 = 0 \\ s_i = s_{i-1} + \Delta s_{i-1} \end{cases} \tag{2}$$

With h_i denoting the weight value, then gray value of each pixel in the image can be obtained by Eq(3).

$$F_o(x, y) = \frac{\sum_{i=0}^{l} F(P_i)\, h_i + \sum_{i=0}^{l'} F(P'_i)\, h'_i}{\sum_{i=0}^{l} h_i + \sum_{i=0}^{l'} h'_i} \tag{3}$$

Here, $F_o(x, y)$ is gray value of the pixel (x, y) in the image after calculation. $F(P_i)$ is the gray value of the pixel P_i in the input noise image. l and l' represent the integral steps of the forward and reverse streamlines respectively. P_i and P_i' are pixel positions of the forward and reverse step i along the streamlines respectively. P_0 is the pixel point (x, y). h_i and h_i' are weights calculated from Eq.(1).

III. STREAMLINE GENERATION MODE

In the line integral convolution-based texture visualization method, the generation of streamlines is very important. This is because the texture values of each sample point are convolved along the vector direction, that is, convolved along the streamline direction. The effectiveness of the streamline is heavily dependent on the choice of the initial seed point and the implementation of the streamline drawing.The effect of streamline drawing depends mainly on the length of the step and whether it is interpolated. There are two main ways to implement streamline drawing: direct pass method and numerical method. The former uses a straight line with the same vector direction as a grid point to draw a streamline through the grid. And the latter uses numerical calculation to approximate the vector values of sampling points (using integration method) and the vector value of the sample point (using interpolation method).

A. NUMERICAL CALCULATION

The streamline equation [16] is defined as shown in Eq. (4):

$$\frac{d\mu(\tau)}{d\tau} = v(\mu(\tau)) \tag{4}$$

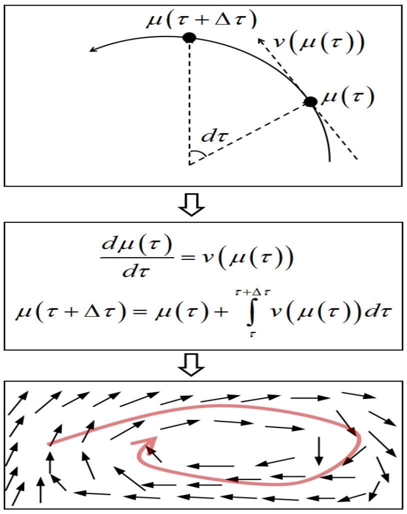

$\mu(\tau+\Delta\tau)$
$v(\mu(\tau))$
$\mu(\tau)$
$d\tau$
$\frac{d\mu(\tau)}{d\tau} = v(\mu(\tau))$
$\mu(\tau+\Delta\tau) = \mu(\tau) + \int_{\tau}^{\tau+\Delta\tau} v(\mu(\tau))d\tau$

FIGURE 3. Streamline generation.

Eq. (4) indicates that the direction of a point on the streamline is the same as the direction of the tangent, where $\mu(\tau)$ represents the position of the point in space, $v(\mu(\tau))$ is the tangent direction of the point on the streamline: v is a function of τ, and τ can be parameters such as time and arc length. The streamline generation process is shown in Figure 3.

The streamline is defined as Eq. (5).

$$\Delta\vec{I} \times \vec{V} = 0 \tag{5}$$

where $\Delta\vec{I}$ is the vector element on the streamline, and $\vec{V}$ is the velocity. Eq.(5) indicates that the tangent direction of any point on the streamline coincides with the velocity direction of the point.

$$\Delta\vec{I} = \Delta X\vec{i} + \Delta Y\vec{j} \tag{6}$$

$$\vec{V} = u\vec{i} + v\vec{j} \tag{7}$$

where u and v are functions of X and Y. Because of $\vec{i} \times \vec{i} = \vec{j} \times \vec{j} = 0$ and $\vec{i} \times \vec{j} = -\vec{j} \times \vec{i}$, we can use equations (6) and (7) into Eq. (4), and thus get Eq. (8)

$$\frac{\Delta X}{u} = \frac{\Delta Y}{v} = k \tag{8}$$

A two-dimensional streamline can be generated by using the Eq. (8).

Eq. (8) can be converted to the following Eq. (9).

$$(\Delta X)^2 + (\Delta y)^2 = k^2\left(u^2 + v^2\right) \tag{9}$$

When u and v are not equal to 0 at the same time, the value of k can be calculated by using the following equation.

$$k = \pm\frac{\sqrt{(\Delta X)^2 + (\Delta Y)^2}}{\sqrt{u^2 + v^2}} \tag{10}$$

When k takes a positive value, then Eq. (8) can be changed to:

$$\begin{cases} \dfrac{\Delta X}{\sqrt{(\Delta X)^2 + (\Delta Y)^2}} = \dfrac{u}{\sqrt{u^2 + v^2}} \\ \dfrac{\Delta Y}{\sqrt{(\Delta X)^2 + (\Delta Y)^2}} = \dfrac{v}{\sqrt{u^2 + v^2}} \end{cases} \tag{11}$$

In Eq. (11), let $(\Delta X)^2 + (\Delta Y)^2 = (\Delta R)^2$, then ΔR denotes the streamline step size that needs to be taken on the streamline. And because $u^2 + v^2 = \left|\vec{V}\right|^2$, then Eq. (11) can be changed to Eq. (12).

$$\begin{cases} \dfrac{\Delta X}{\Delta R} = \dfrac{u}{\left|\vec{V}\right|} \\ \dfrac{\Delta Y}{\Delta R} = \dfrac{v}{\left|\vec{V}\right|} \end{cases} \tag{12}$$

It can be seen that Eq. (12) represents the cosine of velocity in the x, y direction, and the initial condition of Eq. (12) can be written as:

$$\begin{cases} X(\Delta R_i) = X_0 \\ Y(\Delta R_i) = Y_0 \end{cases} \tag{13}$$

(X_0, Y_0) is the starting point of the streamline, which is the initial coordinate of the streamline formed by each line integral convolution.

$$\begin{cases} \dfrac{\Delta X}{\Delta R} = \dfrac{u}{\left|\vec{V}\right|} = f\left(\Delta R, X, Y\right) \\ \dfrac{\Delta Y}{\Delta R} = \dfrac{v}{\left|\vec{V}\right|} = g\left(\Delta R, X, Y\right) \\ X\left(\Delta R_i\right) = X_0 \\ Y\left(\Delta R_i\right) = Y_0 \end{cases} \tag{14}$$

Given the coordinate of the seed point of (X, Y) being (X_0, Y_0), the position of the next point can be obtained by using the fourth-order Runge-Kutta method. A similar analysis can be used for cases where t is negative. If let the integral step size (the distance between the starting point and the next point) be h, then according to the fourth-order Runge-Kutta method, we can obtain the following equations:

$$\begin{cases} K_1 = f\left(\Delta R_i, X_i, Y_i\right) \\ K_2 = f(\Delta R_i + \frac{h}{2}, X_i + \frac{1}{2}hK_1, Y_i + \frac{1}{2}hL_1) \\ K_3 = f(\Delta R_i + \frac{h}{2}, X_i + \frac{1}{2}hK_2, Y_i + \frac{1}{2}hL_2) \\ K_4 = f(\Delta R_i + \frac{h}{2}, X_i + \frac{1}{2}hK_3, Y_i + \frac{1}{2}hL_3) \\ X_{i+1} = X_i + \frac{h}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right) \end{cases} \tag{15}$$

$$\begin{cases} L_1 = g\left(\Delta R_i, X_i, Y_i\right) \\ L_2 = g(\Delta R_i + \frac{h}{2}, X_i + \frac{1}{2}hK_1, Y_i + \frac{1}{2}hL_1) \\ L_3 = g(\Delta R_i + \frac{h}{2}, X_i + \frac{1}{2}hK_2, Y_i + \frac{1}{2}hL_2) \\ L_4 = g(\Delta R_i + \frac{h}{2}, X_i + \frac{1}{2}hK_3, Y_i + \frac{1}{2}hL_3) \\ Y_{i+1} = Y_i + \frac{h}{6}\left(L_1 + 2L_2 + 2L_3 + L_4\right) \end{cases} \tag{16}$$

Thus, based on the Eq.(15) and Eq. (16), and the initial condition (X_0, Y_0), the coordinates (X_{i+1}, Y_{i+1}) below the streamline can be obtained. In regard to the calculation method of the streamline, the Euler method can also be adopted for the requirement of accuracy. Compared with the fourth-order Runge-Kutta method, the Euler method is relatively simple and easy to calculate. In this paper, we use fourth-order Runge-Kutta method to calculate the integral curve.

B. A NEW WAY TO DETERMINE THE INTEGRATION STEP SIZE AND INTERPOLATION METHOD

In a two-dimensional vector field, the input image is a two-dimensional vector image, and a random white noise image of a resolution such as a two-dimensional vector image. One-dimensional box filtering is used as the convolution kernel. The streamline calculation method can use Euler method. The basic idea of Euler method is iteration, and the step size is the edge length of a grid. Regarding the choice of step size, a reasonable step size is critical to the convolution effect. If the sampling distance is too large, the LIC integral will be discontinuous, resulting in aliasing. If the sampling distance is too small, the drawing quality cannot be improved, and the performance is significantly degraded. For random noise, it is obvious that the step size cannot exceed the width of the unit of a pixel. If the sampling distance exceeds this distance, there will be many high frequency components. In this paper, a new improved method for determining the integration step size and interpolation is proposed, which combines the vector speed of the four vertices of the grid where the sample points are located and the angle between the vectors to determine the step size and interpolation.

The diagram of the new method of determining the integration step size and interpolation is shown in Figure 4.

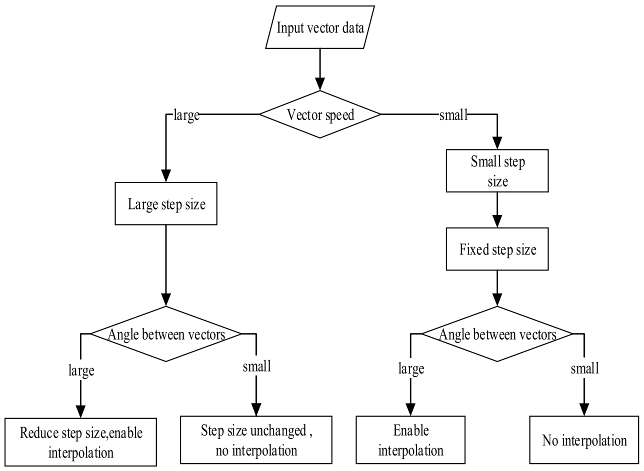


FIGURE 4. Integral step size and interpolation rules.

Let us first look at the vector speed. If the vector speed is small, then the step size is small: if the vector speed is large, then the step size is large. Here, it is necessary to determine whether the angle between vectors of the grid's four vertices is large or not. If the angle is large, then the step size should be reduced; otherwise it will be far from the actual route. As a result, it will result in a streamline that is more consistent than the actual streamline. At the same time, in the case of large vector speed, whether or not interpolation is performed according to the angle between the vectors of the four vertices of the grid where the sampling point is located. If interpolation is not required, the vector of the nearby point can be directly replaced by the vector of the nearby point, reducing the amount of calculation. In addition, when the adjacent angular speed is the same, the integration step size can be extended, possibly exceeding one pixel (here, the length of two adjacent grid points per unit pixel). However, it can't be extended too much (generally no more than two pixels), otherwise it will affect the calculation of texture value.

According to a lot of experiments, we determined that the threshold of judging whether the vector speed is large or small is the average velocity value of the vector field, and the threshold of judging whether the vector angle is large or small is 35°.

C. ANALYSIS OF THE INFLUENCE OF STEP SIZE ON THE STREALINE

Usually the integration step has both a fixed step size and a variable step size. The fixed step size is preferably 0.5 unit pixel, and when the fixed step size is taken as 1 unit pixel, the resulting texture has a large noise. The variable step size mainly determines the length of the step by speed, and the step size is large when the speed is large, and the step size is small when the speed is small. If you encounter a region where the vector direction changes rapidly, this will cause the streamline to deviate from the true trajectory. As the streamline lengthens, the offset becomes larger and larger. The fourth-order Runge-Kutta method can improve the integration accuracy. The fourth-order Runge-Kutta method with variable step size can change the step size by judging whether the integration error of each step is less than the allowable error, thus reducing the truncation error and the accumulated rounding error of each step. However, these required functions have good smoothness. If the function smoothness is not good, then the accuracy of the fourth-order Runge-Kutta method will not be as good as improved Euler method.

The length of integration step plays an important role in streamline drawing. We propose to determine the step size by combining the vector speed with the angle between the two vectors. First, we look at the vector speed. If the vector speed is small, then the step size should be small; if the vector speed is large, then the step size should be large. At this point, it is necessary to see if the angle between the two vectors before and after is large or not. If the angle is large, then the step size should be reduced. Otherwise it will be far from the actual route. In this way, the streamline obtained is more close to the actual streamline.

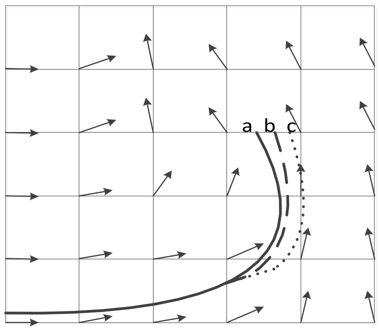


FIGURE 5. Effect of step size on streamline.

Figure 5 shows the streamline of a real trajectory. In particular, streamline **b** is a trajectory of a small step size, and the streamline **c** is a trajectory of a large step size. In the case where only the speed determines the step size, when the angle is large and the speed is small, the influence on the trajectory is small. When the speed is large and the angle is large, the influence is relatively large, and the streamline **c** may occur. Therefore, when the speed is judged, then the angle is judged, and the step size is determined by the speed and angle, the situation of streamline **c** can be corrected. At this time, the streamline generated by the step size is returned to the streamline **b**, which is more suitable for the real trajectory **a**.6

IV. COMPARISON AND ANALYSIS OF VECTOR FIELD VISUALIZATION EFFECTS

The experimental data used in this paper are wind field data, and the size of the data is 370 × 289 vector data. The data value of each grid point contains the vector values of U and V directions. The vector of each grid point can be obtained by combining the vector of U and V directions. The experimental hardware and software environment used in this paper are: Inter(R) Xeon(R) Core(TM2) i7-3615QM CPU 2.40GHz (2 processors), memory 8GB, 64-bit windows operating system, using Visual Studio 2010 and Qt 5.2.0 programming.

Figure 6 shows the results of using LIC algorithm with different integration steps, in the case of no interpolation involved.

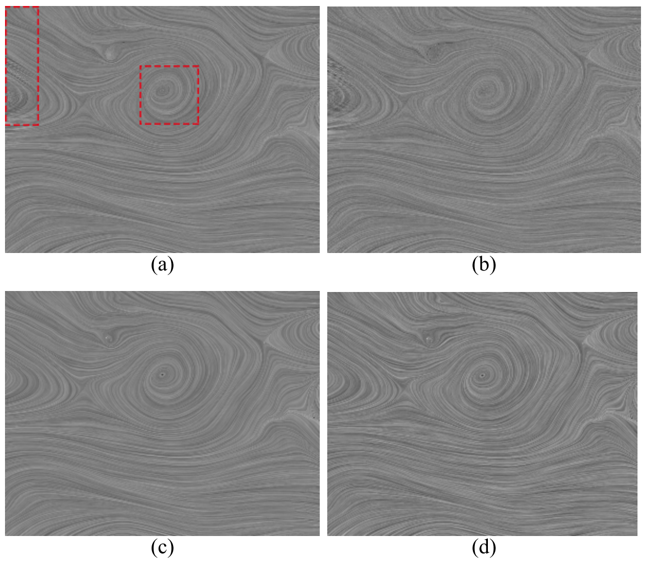


FIGURE 6. Streamline results generated by LIC algorithm with different integration step size. a) Fixed step size is 0.5 unit pixel, b) Fixed step size is 1, c) Step size determined by speed, d) Step size determined by speed and angle.

Figure 6 shows the results of texture visualization with different step sizes. After analysis, it can be seen that in Figure 6(d) the texture of our method is clearer and the contrast is more obvious than that of other methods. The texture of Figure 6(c) and 6(d) is smoother than that of Figure 6a and 6(b), and 6(b) has more noise. The enlargement of the red box position in Figure 6(a) in each comparison images are as following figures:

Figure 7(a)-(d) are enlarged images of the middle red box region shown in Figure 6a-d, and Figure 7(e)-(h) are enlarged images of the upper-left red box region shown in Figure 6(a)-(d). It is clear from the enlarged

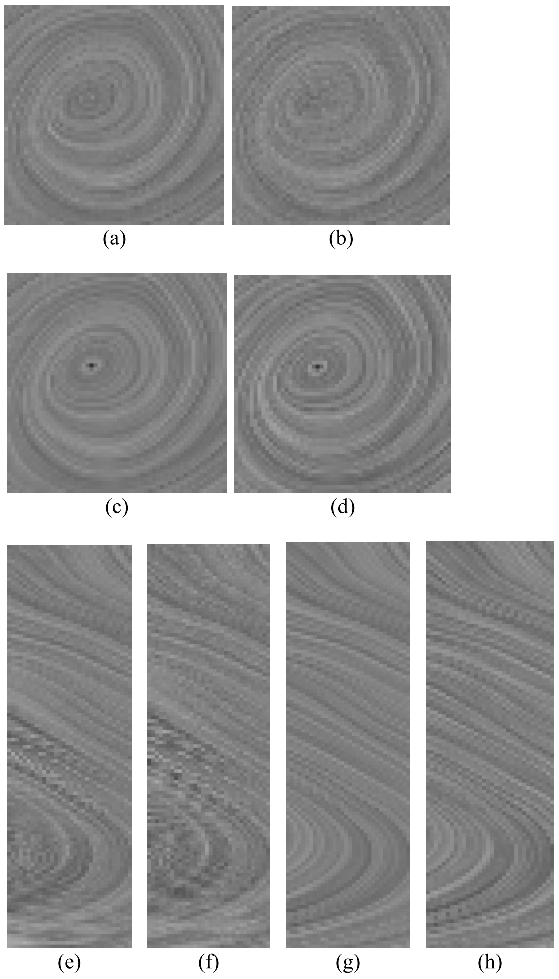


FIGURE 7. Corresponds to the enlarged image of each figure in Figure 6 at the position of the red box in Figure 6a.

Figure 7(d) and 7(h) that the effect of the streamlines by using our method is clearer, and has higher contrast and less noise.

In the case of the same step size in LIC algorithm, we compare the streamline effect and the time consumed by different interpolation methods. It can be divided into the following three cases:

(1) In the case of fixed step size of 0.5 unit pixel in LIC algorithm, the streamlines obtained by different interpolation methods are shown in Figure 8. Figure 8a and Figure 8(b) are streamlines obtained by non-interpolation and bilinear interpolation respectively in the case of fixed step size. Figure 8(c) is streamlines obtained by using vector direction angle to determine the interpolation method in the case of fixed step size. Bilinear interpolation is enabled for large angle, and no interpolation is used for small angle.

In the case of fixed step size of 0.5 unit pixel, different interpolation methods are used. The running time comparison of the three methods in Figure 8 is shown in Table 1.

(2) In the case where the step size is determined by the speed, different interpolation methods are used to generate the streamline results, as shown in Figure 9. The rule of

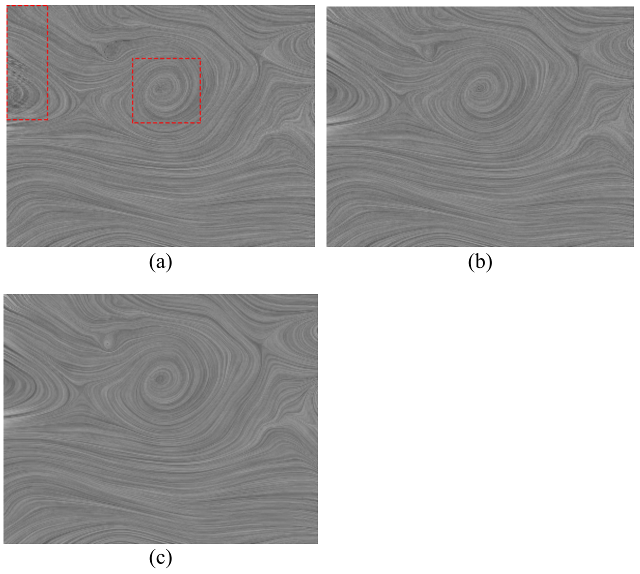


FIGURE 8. Streamlines using different interpolation methods with fixed step size.

TABLE 1. Comparison of different interpolation times with a fixed step size of 0.5.

Interpolation method	No interpolation	Bilinear interpolation	Interpolation determined by angle
Time（s）	0.494789	0.682472	0.524102

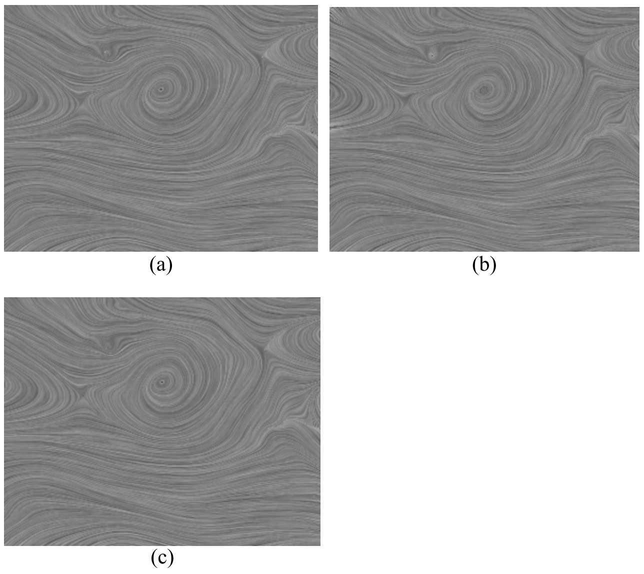


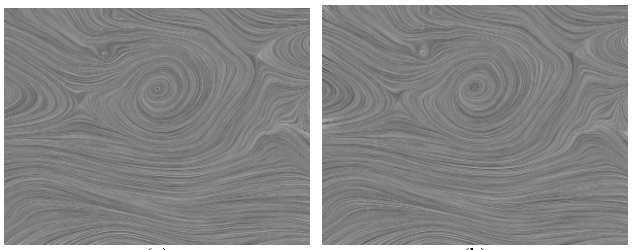
FIGURE 9. Streamlines using different interpolation methods with the step size determined by speed.

determining step size by using speed is that when speed is large, then the step size should be large; when the speed is small, then the step size should be small. Figure 9a and Figure 9b are streamlines obtained by non-interpolation and bilinear interpolation respectively in the case of the step size determined by the speed. Figure 9(c) is streamlines obtained by using vector direction angle to determine the interpolation method in the case of the step size determined by the speed. Bilinear interpolation is enabled for large angle, but no interpolation for small angle.

The running time comparison of the three methods shown in Figure 9 is shown in Table 2.

TABLE 2. Comparison of different interpolation times with a fixed step size of 0.5.

Interpolation method	No interpolation	Bilinear interpolation	Interpolation determined by speed and angle
Time（s）	0.629568	1.192571	0.913515


(a) (b)

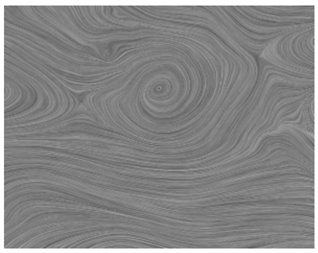

(c)

FIGURE 10. Streamlines using different interpolation methods with the step size determined by speed and angle.

(3) In the case where the step size is determined by both speed and angle, different interpolation methods are used to generate the streamline results as shown in Figure 10. The rules for determining step size are as follows: Firstly, the step size is determined by the speed. When the speed is high, the step size is small. Then, the step size is further adjusted according to the angle of vector, and the step size is reduced appropriately with large angle, and the step size is not adjusted when the angle is small. Figure 10(a) and Figure 10b are streamlines obtained by non-interpolation and bilinear interpolation respectively in the case of the step size determined by the speed and angle. Figure 10(c) shows streamlines obtained by using vector direction angle to determine the interpolation method and using both speed and angle to determine the step size.

The running time comparison of the three methods shown in Figure 10 is shown in Table 2.

In order to analyze the above-mentioned experimental results, we enlarged the experimental results locally according to the position of the red box labeled in Figure 8. In Figure 8, 9, and 10, the local enlargement of the corresponding red box positions is shown in Figures 11 and 12.

TABLE 3. Comparison of different interpolation times with a fixed step size of 0.5.

Interpolation method	No interpolation	Bilinear interpolation	Interpolation determined by speed and angle
Time（s）	0.873130	1.550206	1.184940

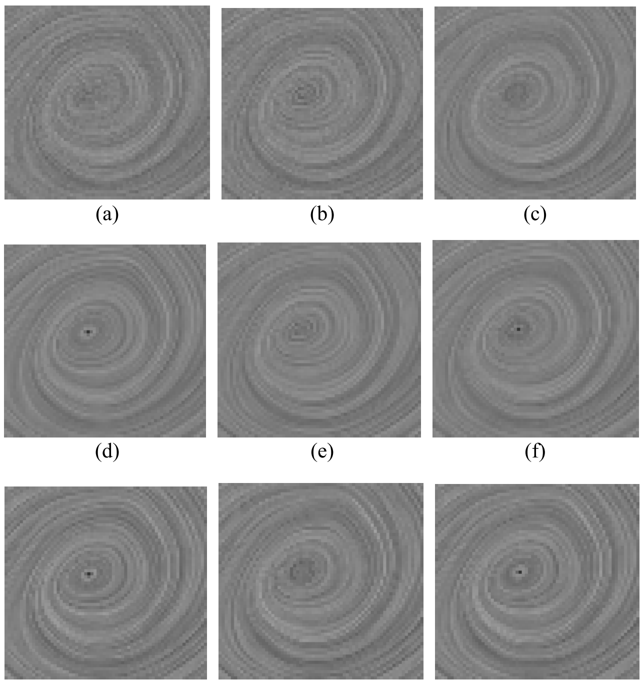
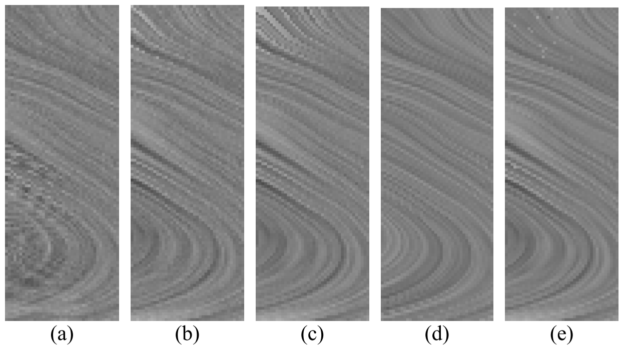

(a) (b) (c)
(d) (e) (f)
(g) (h) (i)

FIGURE 11. Local enlargement of the middle part of Figure 8-10.

In Figure 11, Figure 11(a)-(c) are local enlargement of the corresponding part of Figure 8(a)-(c), Figure 11(d)-(f) are local enlargement of the corresponding part of Figure 9(a)-(c), and Figure 11(g)-(i) are local enlargement of the corresponding part of Figure 10(a)-(c).

From Figure 11, we can see that the streamline obtained by using our method is clearer, and has higher contrast and less noise. Comparing and analyzing the row images in Figure 11: under the condition of identical step size, the proposed method uses speed and angle to determine the interpolation method. Compared with non-interpolation method and bilinear interpolation method, it is clear that the results by using our method has the best effect, such as with the clearest streamline, the strongest contrast and no obvious noise. Comparing and analyzing the column images in Figure 11: in the same case of determining the interpolation mode, the proposed method for determining the step size by the size of speed and angle is better than the fixed step size and only using the speed size to determine the step size. The streamline is clear, the contrast is relatively strong, and the noise is not obvious. The overall effect is from the upper left corner of Figure 11(a) to the lower right corner of Figure11(i), and the streamline is getting better and better.

In Figure 12, Figure 12(a)-(c) are local enlargement of the corresponding part of Figure 8(a)-(c), Figure 12(d)-(f)



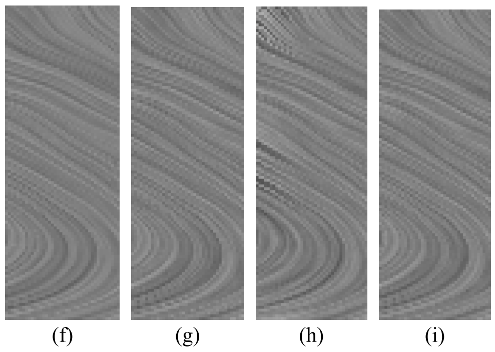


FIGURE 12. **Local enlargement of the upper left corner of Figure 8-10.**

are local enlargement of the corresponding part of Figure 9(a)-(c), and Figure 12(g)-(i) are local enlargement of the corresponding part of Figure 10(a)-(c). Specific analysis is consistent with Figure 11, and the overall effect is getting better and better from Figure 12 (a) to 12 (i). Thus, the new proposed method of determining integration step and interpolation method can get better texture effect.

Like the figures in Figure 11, the figures in Figure 12 are locally enlarged at the upper left corner of Figure 8-10, respectively. The trend is consistent with Figure 11, and is getting better and better from Figure 12(a) to Figure 12(i). The experimental results show that the new proposed integration step size and interpolation determination method can achieve better streamline texture effect.

V. CONCLUSIONS

In the streamline generation of vector field visualization, this paper presents an improved LIC method, which determines the integration step size and interpolation mode of LIC according to the vector speed and angle in vector field. Compared with bilinear interpolation, the proposed method has less time, higher computational efficiency, clear streamline and relatively strong contrast. In the case of identical interpolation methods, the method of determining step size by speed and angle presented in this paper has the best effect, clearest streamline, strongest contrast and no obvious noise. When the step size is determined by both the speed and the angle, and the interpolation method is determined by both speed and the angle, then a clear, noise-free and high contrast texture can be obtained, and the calculation takes less time.

REFERENCES

[1] J. J. van Wijk, "Spot noise texture synthesis for data visualization," *ACM SIGGRAPH Comput. Graph.*, vol. 25, no. 4, pp. 309–318, Apr. 1991.

[2] B. Cabral and L. C. Leedom, "Imaging vector fields using line integral convolution," in *Proc. SIGGRAPH*, Anaheim, CA, USA, 1993, pp. 263–270.

[3] D. Stalling and H.-C. Hege, "Fast and resolution independent line integral convolution," in *Proc. SIGGRAPH*, New York, NY, USA, 1995, pp. 249–256.

[4] M. Zöckler, D. Stalling, and H.-C. Hege, "Parallel line integral convolution," *Parallel Comput.*, vol. 23, no. 7, pp. 975–989, Jul. 1997.

[5] L. K. Forssell, "Visualizing flow over curvilinear grid surfaces using line integral convolution," in *Proc. IEEE Vis.*, Washington, DC, USA, Oct. 1994, pp. 240–247.

[6] V. Interrante and C. Grosch, "Strategies for effectively visualizing 3D flow with volume LIC," in *Proc. IEEE Vis.*, Phoenix, AZ, USA, Oct. 1997, pp. 421–424.

[7] V. Interrante and C. Grosch, "Visualizing 3D flow," *IEEE Comput. Graph. Appl.*, vol. 18, no. 4, pp. 49–53, Jul. 1998.

[8] A. Sundquist, "Dynamic line integral convolution for visualizing streamline evolution," *IEEE Trans. Vis. Comput. Graphics*, vol. 9, no. 3, pp. 273–282, Jul. 2003.

[9] C. Rezk-Salama, P. Hastreiter, T. Christian, "Interactive exploration of volume line integral convolution based on 3D-texture mapping," in *Proc. IEEE Vis.*, San Francisco, CA, USA, Oct. 1999, pp. 233–240.

[10] M. Hlawatsch, F. Sadlo, and D. Weiskopf, "Hierarchical line integration," *IEEE Trans. Vis. Comput. Graphics*, vol. 17, no. 8, pp. 1148–1163, Aug. 2011.

[11] J. Huang *et al.*, "Output-coherent image-space LIC for surface flow visualization," in *Proc. IEEE Pacific Vis. Symp.*, Songdo, South Korea, Feb. 2012, pp. 137–144.

[12] P. Skraba, B. Wang, G. Chen, and P. Rosen, "2D vector field simplification based on robustness," in *Proc. IEEE Pacific Vis. Symp.*, Yokohama, Japan, Mar. 2014, pp. 49–56.

[13] P. Skraba, B. Wang, G. Chen, and P. Rosen, "Robustness-based simplification of 2D steady and unsteady vector fields," *IEEE Trans. Vis. Comput. Graphics*, vol. 21, no. 8, pp. 930–944, Aug. 2015.

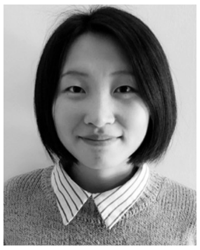
[14] W. Quan, T. Xiaoan, Z. Junda, and K. Longxing, "An approach of vector field texture visualization based on field driven strength," in *Proc. Int. Conf. Inf. Commun. Technol.*, Nanjing, China, 2014, pp. 1–5.

[15] S. Oeltze, D. J. Lehmann, A. Kuhn, G. Janiga, H. Theisel, and B. Preim, "Blood flow clustering and applications invirtual stenting of intracranial aneurysms," *IEEE Trans. Vis. Comput. Graphics*, vol. 20, no. 5, pp. 686–701, May 2014.

[16] J. J. van Wijk, "Image based flow visualization," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 745–754, Mar. 2002.

XUJIA QIN received the M.S. degree in mechanical engineering from Guangxi University, Nanning, China, in 1997, and the Ph.D. degree in mechanical engineering from the Dalian University of Technology, Dalian, China, in 2001. He was a Postdoctoral Fellow with the State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou, China, from 2001 to 2003. He is currently a Professor and a Ph.D. Candidate Supervisor with the Zhejiang University of Technology, Hangzhou. His research interests include computer graphics, image processing, and data visualization.

Note: panel labels (a)–(i) appear under the figures.



XIA FANG received the B.S. degree in computer science from Jishou University, Jishou, China, in 2018. She is currently pursuing the M.S. degree with the Zhejiang University of Technology, Hangzhou, China. Her research interests include computer graphics and digital image processing.

LOUHENG CHEN received the B.S. degree in computer science from the Zhejiang University of Technology, Hangzhou, China, in 2014, where he is currently pursuing the M.S. degree. His research interests include computer graphics and digital image processing.

HONGBO ZHENG received the M.S. degree in geographic information system from Nanning Normal University, Nanning, China, in 2003, and the Ph.D. degree in geographic information system from East China Normal University, Shanghai, China, in 2012. She is currently a Lecturer with the Zhejiang University of Technology, Hangzhou, China. Her research interests include GIS and image processing.

JI MA received the M.S. and Ph.D. degrees in computer science from University College Cork, Ireland, in 2009 and 2014, respectively. He is currently a Lecturer with the Zhejiang University of Technology, Hangzhou, China. His research interests include computer graphics and data visualization.

MEIYU ZHANG received the B.S. and M.S. degrees in control engineering from Zhejiang University, Hangzhou, China, in 1996 and 1999, respectively. She is currently a Professor with the Zhejiang University of Technology, Hangzhou, China. Her research interests include image analysis and image processing.