Evenly-Spaced Streamlines Implementation

Ruoyu Zhuang

Abstract— This paper presents an 2D streamline visualization implementation based on Bruno Jobard and Wilfrid Lefer paper *Creating Evenly-Spaced Streamlines of Arbitrary Density*.[1] This implementation gives users control of the density of the field by allowing users to change the separating distance of streamlines of the image. The quality of the result is as good as other current implementations in this field with fast computation time. Users can also further process the data to generate images with special effect. An example of tapering effect is included in the implementation.

Index Terms—Visualization, streamlines, rendering, flow fields, special effects, vector fields, seeds placement

1 Introduction

VECTOR field visualization is an important technique in scientific research, industrial development and art. It allows people to view and analyse direction and magnitude of different vector fields such as occiean currents, air flow, stress distribution, and thermodynamics. Many methods have been developed dedicated to this visualization. For example, spatial resolution with particle traces, texture based visualization, and image guided visualization. The implementation illustrated in this paper is one of the spatial resolution with streamlines. To get a good image of the vector flow, seed points placement is very important. In this paper, an evenly-spaced seed point selection algorithm applied in the implementation will be presented. It gives users control of the density. The streamlines can be generated in a single run. The rest of the paper is separated into related work in section 2, implementation is section 3 and result and discussion in section 4.

2 Related Work

2.1 Spot Noise

Spot noise is a texture synthesis techinique for vector field visualization introduced by van Wijk. It uses the higher coherency between neighboring pixels to construct streamlines. This method first distributes many spots which are intensity functions over the domain of the data. The function is everywhere zero except for an area that is small compared to the texture size. Spots of random intensity are drawn, and they are rotated and scaled based on underlying vector flow [2]. However, when the flow has a greatly varying gradient, the result in that area may get blur. Willen de Leeuw and Jarke van Wijk introduce a technique called spot blending to address this problem. Spot blending will construct a stream surface by integration of streamlines from a rake perpendicular to the flow in the center of a spot [2]. This technique is computation costly because streamlines have to be traced and a mesh must be rendered for each spot [2]. Thus, tradeoff will be made for using it. Furthermore, the shape of the spots greatly affect the characteristics of the texture [2]. It is not easy to find a spot shape that suits all the vector fields.

2.2 Line Integral Convolution

Line integral comvolution (LIC)[3] is another texture based vector field visualization technique. By using texture, the result image can give a continuous view of 2D field. The algorithm is a function which takes a random input texture and the vector field. Streamlines are calculated from the vector field. An output texture is generated using the streamlines and an one-dimensial local filter. A pixel in the final texture is determined by the weighted sum of a number of pixels along a line in the input texture. It can be used to image dense vector fields in two and three dimensions. With other post-processing modules, LIC texture can have many novel effects [3].

2.3 Image guided method

This method uses an energy function to guide the placement of streamlines at a specified density. The energy function uses a low-pass filtered version of the image to measure the difference between the current image and the desired visual density which is compared against a uniform gray-level [4]. Initially a set of random streamlets are placed. Then the energy function of this image returns an energy value. If the value is too high meaning some of the streamlines in the image are too dense and vice versa. To improve the placement of streamlines, the method changes the positions and lengths of streamlines, joining streamlines that nearly abut, and creates new streamlines to fill sufficiently large gaps [1]. The authors Greg Turk and David Banks wants to achieve results of placing streamlines according to a guiding principle automatically that mimic hand-drawn figures to show the magnitude and direction of vector fields [4]. However, if the vector field is too dense, this method may not converge easily at the end.

3 Methodology And Implementation

3.1 Algorithm Overview

In this implementation, evenly-spaced streamlines vector

• *Ruoyu Zhuang. Author is with the University of Houston, Houston, TX 77201. E-mail: rzhuang@uh.edu.*

field visualization is created by a given file with vector field information, and an initial starting point. When program runs, the coordinates and vectors information of the file are encoded using color information in a 2D texture. Then, a starting point is passed into a function to calculate a streamline by using this point and the information decoded from the 2D texture. The streamline is calculated forward and backward based on the seed points. The sample points generated for the streamline are saved into a 2D grid for later sample points and seed points validation. A streamline contains many sample points. After the first streamline is generated, sample points of this streamline will be used to generate potential seed points based on a user specified separating distance for the next streamline. To generate the potential seed points, two closest sample points are passed into a function. This function finds the perpendicular slope of these two points, and calculate two seed points using this slop and the first sample point. After processing all sample points, an array of all potential seed points from this streamline is generated. Then, potential seed points in this array are going to be validated one by one. If a validated seed point is found, it will be passed into the function to calculating a new streamline. Since this is not the first streamline, all the sample points of this steamline will be validated by calculating the distance between them and sample points from previous generated streamline. If a new sample point cannot pass the validation process because the distance between it and a previous sample point is smaller than the value of separating distance times a factor, the streamline calculation in this direction will strop. Newly generated streamlines are saved in a queue. After all potential seed points having been validated, a streamline is popped out from the queue to generate more potential seed points and streamlines, and the process repeat. When the queue is exhausted, all streamlines should be generated. Following is a piece of sudo code of the algorithm.

```
Main(){
    Seed = Starting point
    Line = CalculateStreamline(Seed)
    Queue.push(Line)

    While(Queue is not empty){

        CurrentLine = Queue.pop
        SeedPoints = CalculateSeedPoints(CurrentLine)

        For(Seed in SeedPoints)
            If Seed is valid
                Line =CalculateStreamline(Seed)
                Queue.push(Line)
    }
}

CalculateStreamline{

     For (i < kernel size)
        Line integration in forward direction
        If the new sample point is valid
            Save it into this streamline and Bin grid
        Else
            Stop integration in this direction
    For (i < kernel size)
        Line integration in backward direction
        If the new sample point is valid
            Save it into this streamline and Bin grid
        Else
            Stop integration in this direction

}

ValidSamplePoints{

    For (point in SamplePoints)
        Calculate corresponding Bin coordinates of the Point
        Calculate distance between the Sample Point to the local Bin and 8 other surrounding bins.
        If there is not exit a sample point that has a distance to the point smaller than the value of separating distance time factor
            Return valid
        Else
            Return not valid

}
```

To reduce computation cost, not all distances between sample points and seed points need to be calculated. The only distance need to be computed is between a point and potential points about it. A special 2D grid is used to save all the validated sample points. This 2D grid is constructed using bin size. For example, if the image has a resolution of 512 * 512. With a user defined separating distance of 8, the bin size will be set at 512 / 8 = 64. Thus, the 2D grid has a size of 64 * 64. A sample point is saved into this grid based on its coordinated value regarding to the bin size. For example, if a sample point of a streamline has x coordinate at 88 and a y coordinate at 112, it will be save into the 2D grid at 88 / 8 =11, and 112 / 8 = 14. With this 2D grid, when a sample or seed point needs validation, it only needs to be compared with the points in its own bin and other eight bins around it. In this way, computation time is greatly reduced.

For line intergration, I used Euler method with a factor of 0.5. This method generates a good result and it is easy to implement. There are other methods such as Eunge-Kutta, and Midpoint methods that are worth investigating in the future to reduce computation error.

In order to have a smooth and long streamline, a factor is used to validate new sample points. The new separating distance to computer sample points, called distance test, is the value of the user defined distance times a factor from 0 to 1. If the distance between a sample point and all existing sample points of different streamlines is bigger than the distance test, then this sample point is valid.

3.2 Tapering Effect

Tapering effect makes the final streamlines look like that

they are drawn by a pen. It is achieved by changing the thickness of each line segment of a streamline. If the smallest distance between a sample point and other sample points of different streamlines around it is bigger than the user defined separating distance, an initial thickness value is applied to this line segment. Otherwise, this thickness of the line segment is calculated by the following formula

$$\frac{smallest\ distance - distance\ test}{separating\ distance - distance\ test} * initial\ thickness\ value$$

4 Results And Discussions

Some generated vector field image with a resolution of 512 * 512 using file *bnoise.ply* are shown below with their separating distances, factors and starting positions used for calculating streamlines.

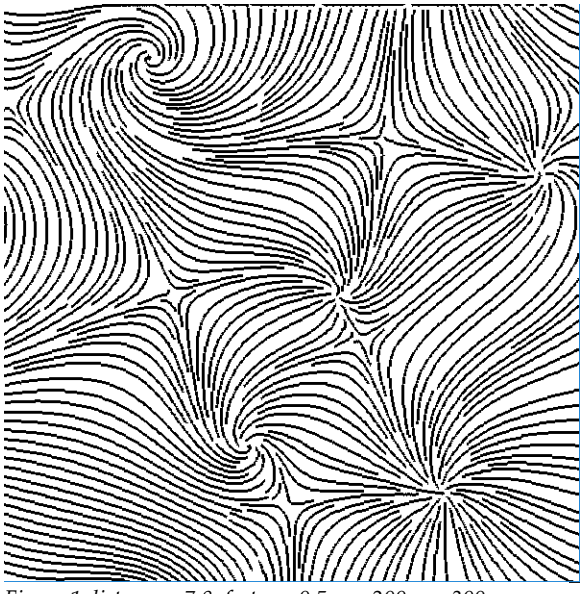

Figure 1 distance = 7.8, factor = 0.5, x = 200, y = 300

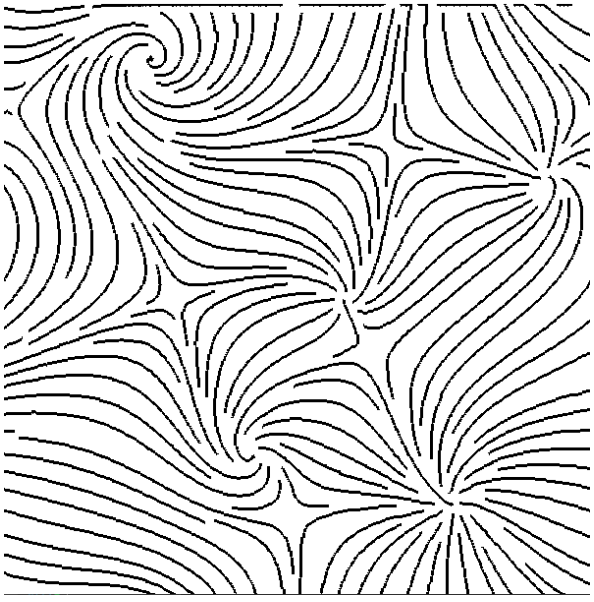

Figure 2 distance = 15, factor = 0.46, x = 200, y = 300

Figure 1, 2 and 3 are images with a separating distance of 1.5 %, 3% and 5.6% of the total image size. The image quality varies with different value of parameters. So it is important to fine tuned the parameters to get a better result. The streamlines of figure 1 are long and almost evenly spaced with less separation. As the separating distance becomes larger, some streamlines become shorter and some streamlines are missing. Especially around edges and concentration areas. For starting positions, as long as they are not in the edge of the image, I did not observe much quality interference. The streamlines with tapering effect shown in figure 4 still have some artifacts. I believe the reason is that the smallest distances between a sample points of a steamline and sample points of other streamlines are not uniform. Some are smaller and some are larger.

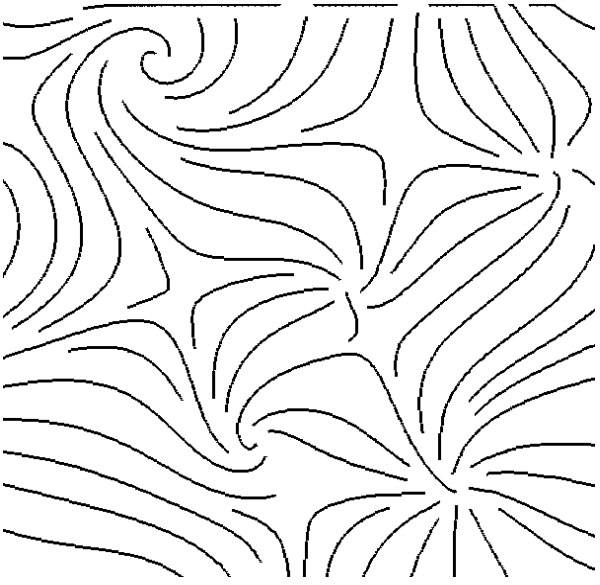

Figure 3 distance = 29, factor = 0.41, x = 200, y = 300

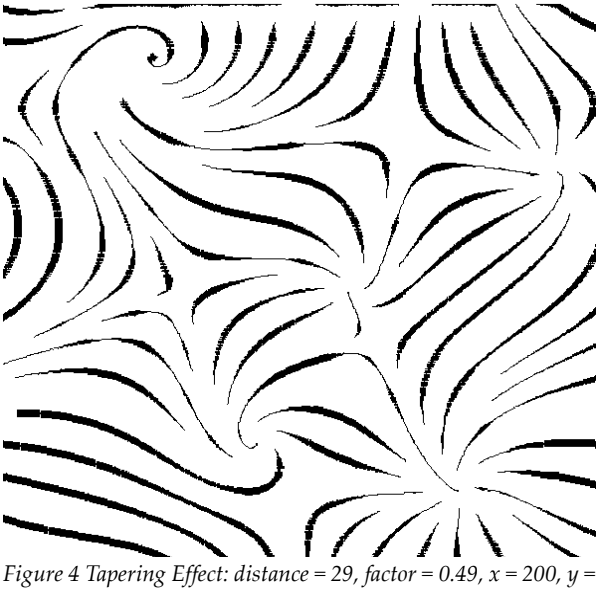

Figure 4 Tapering Effect: distance = 29, factor = 0.49, x = 200, y = 300

5 Conclusion and Future Work

The implementation result shows that the method developed by Bruno Jobard and Wilfrid Lefer [1] is very effective. It can generate image in a single run and computation time is less than 1 second without sacrificing image quality. Compared to other methods such as LIC and image guided vector field visualization, this implementation gives user easy control of the streamlines density, and an ability to easily further process result to generate other effects. There are still directions and methods that are worth investigating in the future. A better line integration method such as higher order Runge-Kutta method may be used to reduce errors and artifacts, so that the streamlines are better sampled and the image is smoother for viewers. Increase Bin 2D grid resolution to futher reduce compuatation cost. Furthermore, in current algorithm, magnitude information is not explicity shown. A better algorithm can be developed to futher adjust user defined distance to show magnitude information among streamlines.

Acknowledgment

The author wish to thank Dr. Chen Guoning for his patient and inspirational guidance during the work.

References

[1] Bruno Jobard and Wilfrid Lefer, "Creating Evenly-Spaced Streamlines of Arbitrary Density," *1997*

[2] W. C. de Leeuv and Jarke J. van Wijk "Enhanced Spot Noise for Vector Field Visualization". *Proc. Of Visualizations*, pp. 233-239, IEEE Press, Los Alamitos, CA, 1995.

[3] Brian Cabral and L. Leedom, "Imaging Vector Fields Using Line Integral Convolution," *Computer Graphics*, pp. 27:263-272, 1993.

[4] Greg Turk and David Banks, "Image-guided Streamline Placement." *Computer Graphics*, pp.30:453-460, jul 1996.