Fast Visualization of 2D Dynamical Systems by the use of Virtual Ink Droplets

Helwig Löffelmann Andreas König Eduard Gröller

{helwig|koenig|groeller}@cg.tuwien.ac.at
Institute of Computer Graphics, Vienna University of Technology,
Karlsplatz 13/186/2, A-1040 Wien, Austria,
http://www.cg.tuwien.ac.at/home/

Abstract. This paper presents a new visualization technique for flow fields in 2D. It utilizes a physical model of smearing ink over a sheet of paper as an intuitive metaphor for the representation of a dynamical system. This technique is capable of producing images that are comparable to those generated with line integral convolution (LIC), which is a well-known and established visualization technique for planar vector fields. Similar to oriented line integral convolution (OLIC), an extension to LIC, the virtual ink droplet method is capable of visualizing not only direction and velocity of flow (as LIC does), but also the orientation of vectors. The main advantage of the new method is, that animation sequences, which intuitively represent the dynamics induced by the underlying dynamical system, can be computed much more efficiently than by the use of LIC or OLIC. A speed-up of about 200 is usually achieved when virtual ink droplets are used instead.

Keywords: visualization, dynamical systems, flow visualization

Introduction

During the past decade scientific visualization emerged from computer graphics as an important separate field of research. Applications of visualization can be found, for example, in medical imaging, database exploration, and 3D flow visualization [3][5].

Flow visualization or, in more general terms, the visualization of dynamical systems, is an important subtopics of visualization [2][7][8][14]. Vector fields that result from numerical simulations, experiments, or measurements, and analytically specified dynamical systems, which are designed to be models of real world phenomena, are visualized. Often the data, which is to be visualized, is given in either 2D or 3D. Therefore most visualization techniques in this field are dealing with two or three dimensions.

Especially in 2D many visualization techniques for vector fields have been developed in the past years. Hedgehogs, i.e., plotting tiny arrows, and streamline representations [6] were followed by more sophisticated techniques as, e.g., spot noise [10] or line integral convolution (LIC) [4]. Both spot noise and line integral convolution are capable of representing the entirety of continuous flow data in 2D within a single image. Direction and velocity of the flow can be visualized at all points of the domain simultaneously. See Figure 1 for two vector fields visualized by the use of LIC.

Although line integral convolution as well as spot noise are able to represent the direction and velocity of 2D flow quite good, both methods have a major drawback: the orientation of flow is not encoded within a single image. To overcome this limitation OLIC (oriented line integral convolution) was developed [13]. A jittered set of bright dots, that are placed on a dark background, is used instead of white noise as texture for a LIC-like convolution operation. Thereby images can be generated, which consist of comet-like icons representing the flow similarly as with spot noise or LIC, but which simultaneously encode the orientation

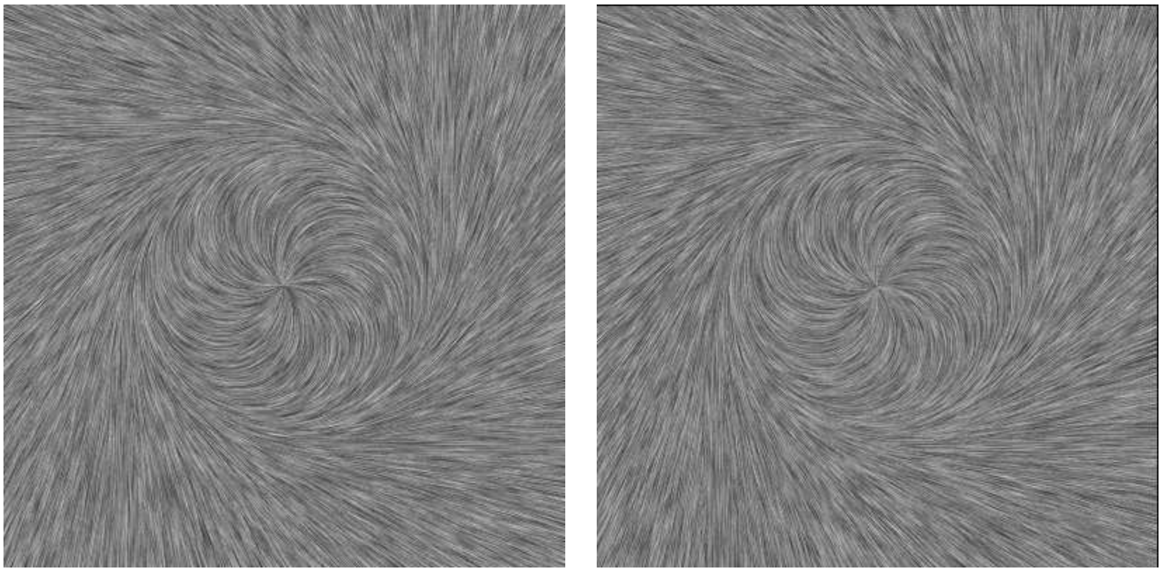


Figure 1: Two vector fields with opposite flow orientation containing a fixed point and a cycle each visualized by the use of LIC. The orientation of flow is not encoded in the images.

of flow. See Figure 2 for a OLIC visualization of the same vector fields as visualized in Figure 1. The difference between both data sets, namely the opposite orientation, becomes obvious by the use of OLIC.

Unfortunately the computation of spot noise, LIC, and OLIC is very time-consuming, because a costly convolution operation is computed for every pixel of the final image. Therefore Fast LIC was introduced to overcome this limitation [9]. Exploiting the spatial coherence in the computation of neighboring pixels on a streamline, the calculation can be accelerated by an order of magnitude as compared to LIC. Some limitations as, e.g., a limited set of possible convolution kernels, are introduced. Similar to LIC and Fast LIC, OLIC was improved by the introduction of FROLIC (fast rendering of oriented line integral convolution) [12]. Disks of small extent are plotted consecutively along short streamlets to approximate the OLIC convolution operation. This allows to effectively draw images that are quite similar to OLIC results.

In addition to all the previously mentioned techniques for the visualization of 2D dynamical systems, we present a new and efficient visualization method, which is similar to OLIC, but with much shorter calculation times. The principal idea of virtual ink droplets is based on an effect that is well-known from school: ink is smeared over the paper due to carelessness. What is usually disliked in school can be exploited for an intuitive visualization of 2D dynamical systems. Virtual ink droplets allow a fast visualization of 2D dynamical systems. In contrast to OLIC and FROLIC physical parameters like paper roughness are specified which increase the flexibility of the method. See also a web-page presenting some results (images and animations) via URL `http://www.cg.tuwien.ac.at/research/vis/dynsys/InkDroplets97/`.

1 Virtual Ink Droplets for the Visualization of Dynamical Systems

Moving the hand over a paper where something has been written onto by the use of ink will result in smeared ink all over the paper. The movement of the hand can be interpreted as a dynamical systems or vector field. The smearing of the ink due to the movements of the hand is a visualization of this dynamical system. Since this association is quite intuitive, it is a useful idea to use this metaphor for developing a new visualization technique.

Simulating the process of smearing real-world ink droplets which initially are spread over a sheet of paper requires a physical model of ink droplets, i.e., virtual ink droplets. Since smearing is an incremental process over time in real world, the model of virtual ink droplets is similarly used to create animation sequences

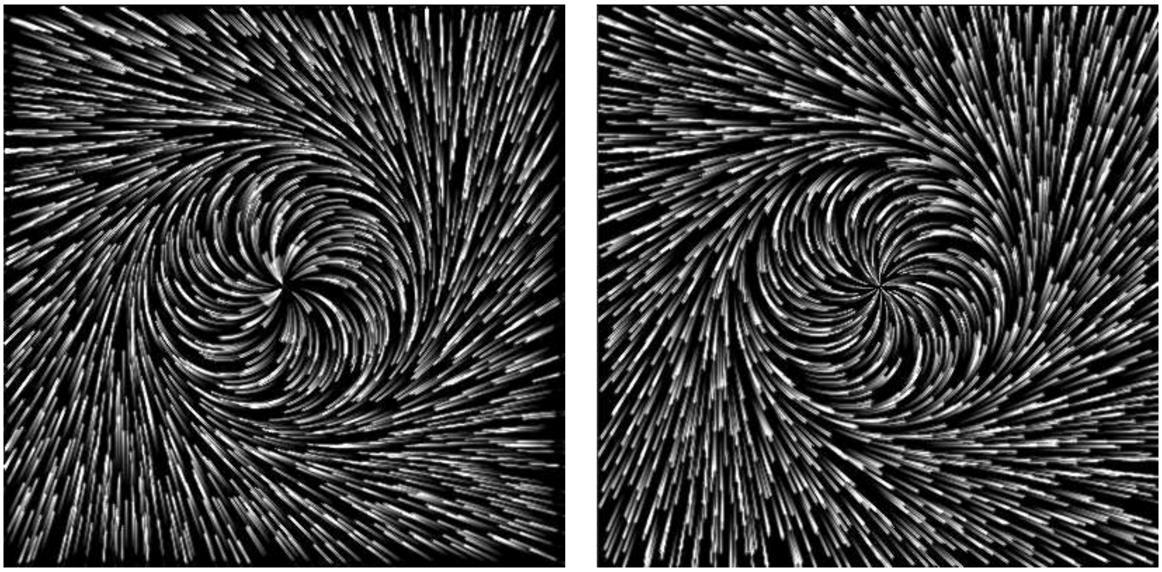
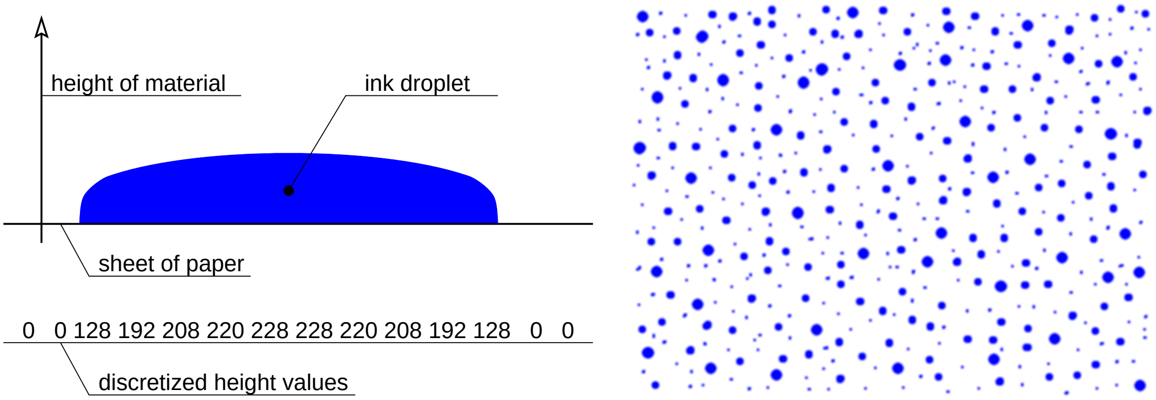


Figure 2: Two vector fields with opposite flow orientation containing a fixed point and a cycle each visualized by the use of OLIC. The orientation of flow is visualized in the images also.

instead of single images. The use of an intuitive metaphor in combination with animation techniques improves the understandability in comparison to other methods even further. Similar techniques, i.e., taking a 2D concentration field, followed by advection of this field, have been studied in the CFD literature [11].

1.1 Modeling Virtual Ink Droplets

The key aspect of modeling real-world ink on a sheet of paper is to approximate the material placed on the paper as a height field. Thereby a type of paper is approximated, which does not suck up the ink immediately. Due to cohesion within the ink and adhesion at the contact between ink and paper a curved cross-section as shown in Figure 3 seems to be a useful approximation. Spreading the ink over the paper is done by the use of a jittered grid and virtual ink droplets with random variation in size. See Figure 3 for the resulting input



(a) (b)

Figure 3: Modeling ink as a height field (a) and resulting texture input (b)

texture. To improve the realistic impression of animation sequences, the roughness of the paper is modeled as varying ink absorption of the paper. A randomly perturbed average value was found to be sufficient in most cases.

1.2 Efficient Calculation of Animation Sequences

An important aspect of this new visualization method is the efficient calculation scheme. It is based on four arrays representing the input to the algorithm:

- The dynamical system to be visualized is specified as a uniformly sampled array V of 2D vectors. For each pixel in the final image one sample V [u,v] is needed by the algorithm.
- The set of virtual ink droplets is represented by a height field I. For each pixel in the resulting image one height value I [u,v] is required.
- Another input to this method is an array A, which describes the roughness of the paper. This array specifies the amount of ink, which can be absorbed by individual pixels.
- A background image B is also used by this method. Similarly to the other input textures B is specified for all pixels of the output image.

The various parameters necessary for the algorithm increase its flexibility. The roughness of the paper may, for example, be used to identify interesting portions of the flow field. The algorithm to produce the animation sequence is quite simple. It proceeds through the animation frame by frame. The ink is propagated over the paper due to the underlying dynamical systems V. Step by step parts of the ink represented in height field I are absorbed by the paper according to its roughness A. Remaining ink is advanced to be processed during consequent steps. To encode the status of ink (processed vs. to be processed), a second height field J is used for ink saved for the next step. After the completion of one frame array J is assigned to be input I for the next step.

The handling of two height fields (I, J) can be implemented efficiently without any need to copy the entire height field after each frame. A time stamp is used for each value in the height field, which records the frame it was written. Before any value is used from the height field, its time stamp is compared with the current frame number to omit all those values which were not written during the last pass. Thereby areas without any ink are automatically skipped during the calculation. This again reduces the calculation time significantly as the amount of disposable ink decreases during the animation.

To obtain the position where ink is moving during the smearing process, vector field V must be integrated numerically. Location values at sub-pixel resolution must be considered to provide a reasonable accuracy of the flow field integration. An internal array H is used to locally store the intermediate positions with floating point precision that are computed during the streamlet integration. Due to performance reasoning just one position data H [u,v] is maintained for each pixel. Although this assumption is not valid for all possible cases—more than one streamlet may enter pixel [u,v] and thereby would require multiple entries in H [u,v]—results show that errors introduced thereby are negligible.

After calculating the propagation of ink during one time step, the resulting output O is merged with the background B. This is done using α-blending and interpreting the amount of ink, that was absorbed during previous steps, as opacity. Altogether the algorithm can be summarized as follows:

```
for all indices [u,v] do:  O[u,v] := 0        {initialize output image}
for each frame i do:
  for all indices [u,v] of I do:
    if I[u,v].time == (i-1) then do:                   {check timestamp}
      O[u,v] += I[u,v].ink*A[u,v]                           {absorb ink}
      [r,s] := integrate(u,v,tau)          {where does the ink go?...
                                                   ...update array H}
      J[r,s].ink  += I[u,v].ink*(1-A[u,v])         {push ink forward}
      J[r,s].time := i                             {update timestamp}
   swap I,J logical
   merge O with B                                       {final image}
   output O
```

Function `integrate(u,v,tau)` is used to advance the integration of streamlets. $H[u,v]$ is used as an initial 2D condition for a numerical integration of V over a time period of `tau`. The result of this integration step is again stored in H, namely in $H[r,s]$, whereby $[r,s]$ denotes the index of the pixel containing $H[r,s]$. According to step size `tau` and the flow field V in the vicinity of $[u,v]$, three different cases can occur:

- $[r,s]=[u,v]$: ink is propagated just a small step within a pixel.
- $[r,s]$ is a neighbor of $[u,v]$: ink is propagated to an adjacent pixel.
- $[r,s]$ is not a neighbor of $[u,v]$: some form of a Bresenham-like algorithm would be necessary to consistently contribute ink to all pixels in-between $[r,s]$ and $[u,v]$. Using small values of `tau` this case becomes very rare and can be ignored.

As already mentioned, the use of array H (one entry per pixel) sometimes produces cases, where the integration of flow V causes more than one streamlet to enter one single pixel. This usually happens in regions of significant convergence and results in a merge of both streamlets into a single one. Thereby some small additional error is introduced, but results show that these inaccuracies are acceptable.

2 Results and Comparisons

In the following the capabilities of the virtual ink droplet method are briefly illustrated by a few sample images. All the pictures presented here were calculated on a SGI Indy workstation by the use of a testbed, which was implemented on the basis of AVS [1], a general purpose visualization system, which is available for multiple platforms. Furthermore the virtual ink droplet method was implemented on a PC using C. Assembler was used to optimize the inner loop of this method.

A major reason for the use virtual ink droplets is the high intuitivity of the resulting images, several possibilities were considered to further enhance this effect. For example, a simulation of squared paper was used as a background for resulting images. Viewing this background well-known from school the user automatically associates the visualization and the ink droplet metaphor. This consecutively helps to interpret the visualization. See Figure 4 for two images demonstrating this effect.

Figure 5 illustrates a dynamical system, which contains a non-linear saddle fixed point. Figure 6 represents one of the vector fields shown in Figures 1 and 2. The dynamical system exhibits an attracting node right in the middle, surrounded by an instable cycle.

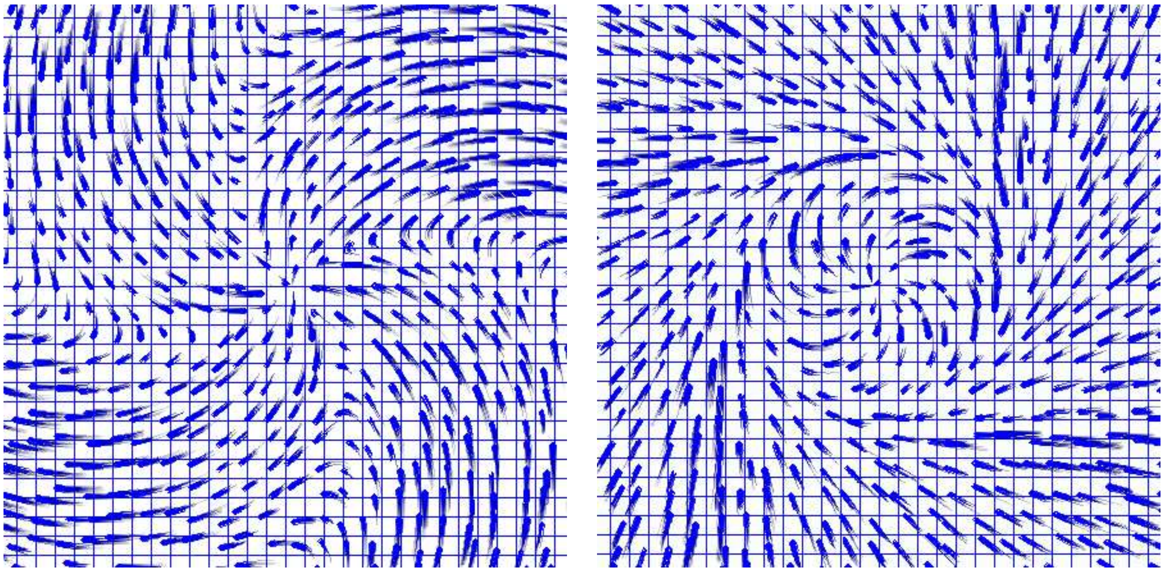
For both images the texture shown in Figure 3b was used as the initial set of ink droplets. Squared paper was used as a background. Flow direction, velocity, and orientation can be easily inferred from the images. The calculation time for both images was about two magnitudes less than for the corresponding OLIC images.

In addition to OLIC-like images as shown in Figure 4 pictures of various other types are possible by the use of virtual ink droplets. Figure 5, for example, shows a LIC-like image. Paper roughness texture A was used to add further information. The absorption coefficient was reduced at the location of the axes and labels, such that almost no ink is absorbed in these places. This can be seen as a simulation of writing text by the use of wax onto the squared paper background.

In Figure 5 a vector field is visualized which exhibits two repelling fixed points, that are both characterized by spiraling local behavior, and a saddle fixed point in-between. The same dynamical system was visualized in Figure 6. Sparsely distributed ink droplets of initially differing size were used as input texture. The visualization is structured into a few rather big droplets, which provide a rough overview over the system behavior. Smaller droplets fill up the spaces and complete the visualization by representing further details.

3 Implementation

The virtual ink droplet method was implemented on two different platforms. First it was implemented within the visualization system AVS [1]. All textures can be easily provided to our method using the data flow concept of AVS.



(a) (b)

Figure 4: Two dynamical systems visualized by the use of the virtual ink droplet method. Squared paper was used as background.

The second implementation was done on a PC platform (DX 2/66). Memory access and hardware graphics output was tuned to maximum speed. Computational intensive parts were written in Assembler. The frame rate of calculated animation sequences at a resolution of 320 by 200 was about 10 frames per second. Therefore the virtual ink droplet technique can be used even if real-time feedback is required.

4 Conclusions

The virtual ink droplet method has been presented as an extension to LIC and other related visualization techniques. An intuitive metaphor for the dynamics induced by a dynamical system in 2D is established by simulating the process of smearing ink droplets over a sheet of paper. A physical model of real-world ink droplets and the smearing process is presented as a basis for this new visualization technique.

By the use of the virtual ink droplet method a rich variety of images can be produced. LIC-like images are possible as well as those usually produced with OLIC or FROLIC. The main advantage of the technique proposed in this paper is that animation sequences can be produced in an efficient way as compared to LIC or OLIC. The calculation time for the virtual ink droplet method is about two orders of magnitude faster than for LIC or OLIC.

Virtual ink droplets are an appropriate tool to efficiently visualize dynamical systems in 2D. It is intended to produce animation sequences which intuitively represent the dynamics induced by the underlying data. The visualization algorithm is flexible with respect to the format of input data. It is possible to use either sampled data or analytically specified dynamical systems for the visualization. Ideas for future work include the use further parameters of the algorithm, e.g., the color of ink or the size of initial ink droplets, for the visualization of additional parameters of the flow data.

5 Acknowledgements

The authors would like to thank Rainer Wegenkittl, who contributed to this work as a continuous and critical observer.

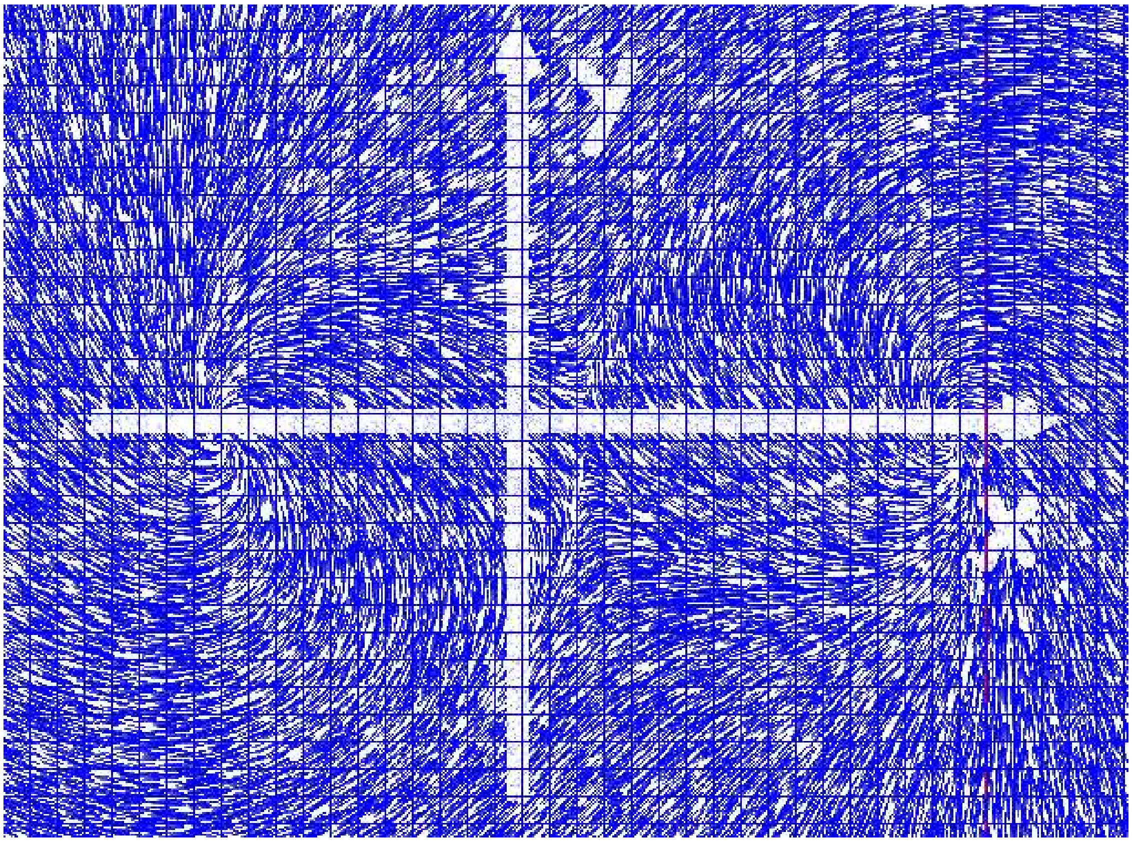


Figure 5: A dynamical system with three fixed points visualized by the use of the virtual ink droplet method.

References

[1] Advanced Visual System Inc. *AVS Developers Guide, Release 4*, May 1992.

[2] H. Aref, R. D. Charles, and T. T. Elvins. Scientific visualization of fluid flow. In C. A. Pickover and S. K. Tewksbury, editors, *Frontiers of Scientific Visualization*, pages 7–43. 1993.

[3] K. W. Brodlie, L. A. Carpenter, R. A. Earnshaw, J. R. Gallop, R. J. Hubbold, A. M. Mumford, C. D. Osland, and P. Quarendon, editors. *Scientific Visualization: Techniques and Applications*. Springer, 1992.

[4] B. Cabral and L. C. Leedom. Imaging vector fields using line integral convolution. In *Proceedings SIGGRAPH '93*, page 263ff, 1993.

[5] R. A. Earnshaw and D. Watson, editors. *Animation and Scientific Visualization: Tools and Applications*. Academic Press, 1993.

[6] C. Hansen. Visualization of vector fields (2d and 3d). In *SIGGRAPH '93, Course Notes no. 2, Introduction to Scientific Visualization, Tools and Techniques*, August 1993.

[7] F. H. Post and T. van Walsum. Fluid flow visualization. In H. Hagen, H. Müller, and G. M. Nielson, editors, *Focus on Scientific Visualization*, pages 1–40. Springer, 1993.

[8] F. H. Post and J. J. van Wijk. Visual representation of vector fields: Recent developments and research directions. In Rosenblum et al., editor, *Scientific Visualization – Advances and Challenges*, pages 367–390. Academic Press, 1994.

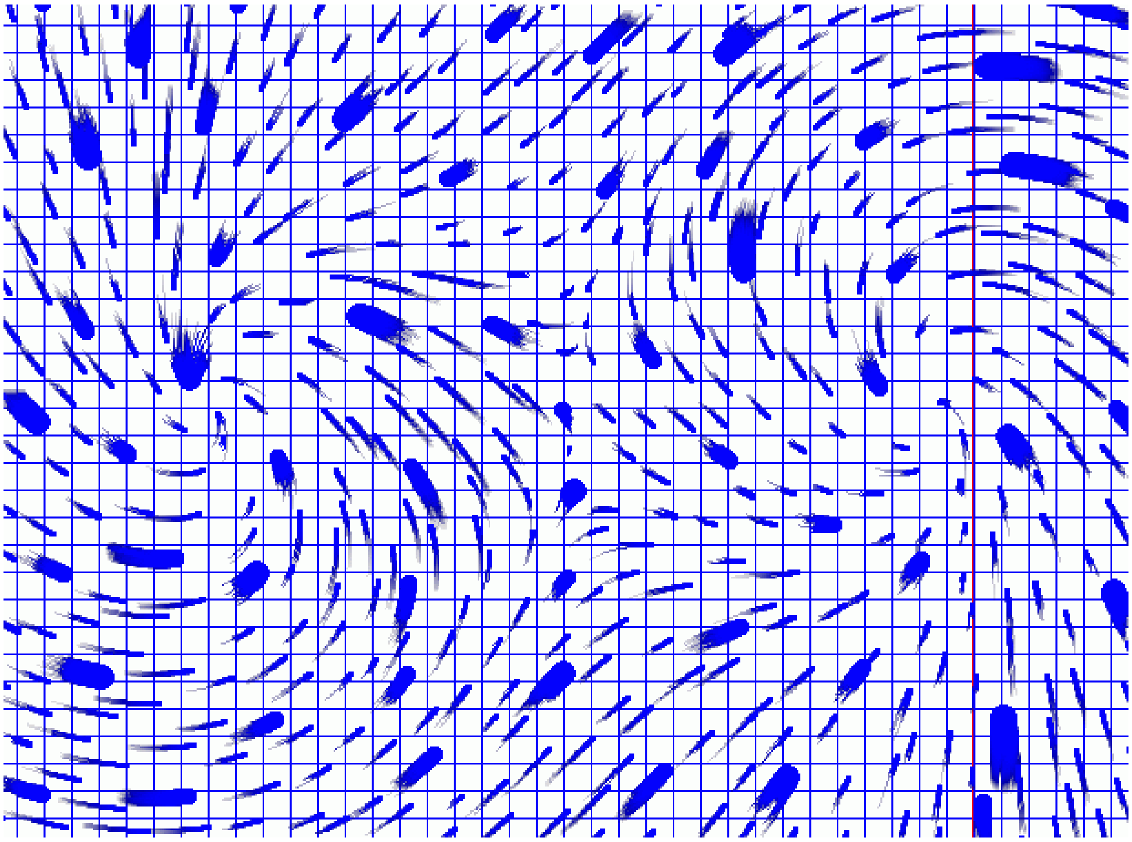


Figure 6: Another image of the same dynamical systems as visualized in Figure 5

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

[10] J. J. van Wijk. Spot noise: Texture synthesis for data visualization. In *Proceedings SIGGRAPH '91*, pages 309–318, July 1991.

[11] J. J. van Wijk. Implicit stream surfaces. In *Proceedings Visualization '93*, pages 245–252. Netherlands Energy Resaerch Foundation ECN, 1993.

[12] R. Wegenkittl and E. Gröller. Fast oriented line integral convolution for vector field visualization via the internet. Technical Report TR-186-2-97-07, Institute of Computer Graphics, Vienna University of Technology, March 1997.

[13] R. Wegenkittl, E. Gröller, and W. Purgathofer. Animating flowfields: Rendering of oriented line integral convolution. Accepted for publication in the Proceedings of the *Conference* Computer Animation, Geneva, Switzerland, June 5-6, 1997.

[14] M. Zöckler, D. Stalling, and H.-C. Hege. Interactive visualization of 3d-vector fields using illuminated stream lines. In *Proceedings Visualization '96*, pages 107–113, 1996.