Visualization of Analytically Defined Dynamical Systems

Eduard Gröller, Helwig Löffelmann, and Rainer Wegenkittl

Vienna University of Technology, A-1040 Vienna, Austria

Abstract. The visualization of analytically defined dynamical systems is important for a thorough understanding of the underlying system behavior. An overview of theoretical concepts concerning analytically defined dynamical systems is given. Various visualization techniques for dynamical systems are discussed. Three current research directions concerning the visualization of dynamical systems are treated in more detail. These are: texture based techniques, visualization of high-dimensional dynamical systems, and advanced streamsurface representations.
Keywords: dynamical system, visualization, texture, multidimensional data, parallel coordinates, streamsurface

1 Introduction to Analytically Defined Dynamical Systems

A dynamical system is a system whose temporal evolution from some initial state is dictated by a set of rules. Dynamical systems are found in many fields of research. Some examples are fluid flow analysis, economic processes like stock market models, physics, medicine, and population growth models [AP90]. Visualizing the behavior of dynamical systems is crucial for a deeper understanding of the underlying dynamics. Dynamical systems are either given as an analytical specification or as sampled data. In the following we will concentrate on analytically defined dynamical systems.

Figure 1 shows a general definition of a dynamical system. The state of the system at a specific point in time is given by the vector $x = (x_1 x_2 \cdots)^T$ of state variables. Depending on the current state x and input u the future evolution of the system as well as output y is determined. The behavior of dynamical systems can be investigated in *phase space*, where each state variable corresponds to a coordinate axis. A point in phase space completely describes the state of the system at one point in time.

Autonomous systems are characterized by the fact that input and output are omitted from the definition. Dynamical systems are either *continuous* or *discrete*. Continuous systems (also denoted flows or vector fields) are given by a set of differential equations $\dot{x} = \nu(x)$. Vector $\nu(x)$ describes the direction, orientation and velocity of the flow at position x. Discrete dynamical systems (often called maps) are specified by a set of difference equations $x_{n+1} = \nu(x_n)$. The temporal evolution from an inital state is called *trajectory* in case of a continuous system and *orbit* in case of a discrete system.

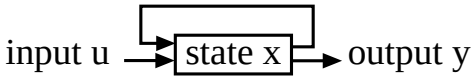


Fig. 1. Specification of a dynamical system

Function $\nu()$ may also contain constants, so called system parameters. A simple example of a one-dimensional discrete dynamical system is the logistic equation $x_{n+1} = f(x_n) = ax_n(1-x_n)$, where a is the only system parameter. Varying parameter a may produce dynamical systems with greatly differing long-term behavior.

An important criterion for the analysis of a dynamical system is whether it is time-dependent or not. For time-dependent dynamical systems function $\nu()$ depends on time whereas for time-independent systems function $\nu()$ does not change over time. Another important characteristic of a dynamical system is whether it is *linear* or not. Linear dynamical systems are rather simple to analyse as opposed to non-linear systems, which typically do have intricate dynamical behavior [Tso92]. Often linearization is used to investigate these complex non-linear dynamical systems at specific locations in phase space.

Hyperbolic dynamical systems are structurally stable, i.e., small perturbations of the system parameters do not change the qualitative behavior of the system. Hyperbolic dynamical systems can be often analysed efficiently by linearization. *Non-hyperbolic* dynamical systems on the other hand are difficult to investigate, occur rarely, and are often the transitional phase between two hyperbolic systems with qualitative differing behavior (bifurcation scenarios).

Starting from an initial state x_0 the solution (i.e., trajectory) of a continuous dynamical system is given as curve $x(t)$ in phase space (see Equation (1)).

$$x(t) = x_0 + \int_0^t \nu(x(u))\, du \tag{1}$$

Equation (1) is an integral equation which can be solved analytically only in very simple cases. Typically numerical integration is used to determine the trajectory of such a dynamical system. Thereby the continuous system is approximated by a discrete dynamical system. The simplest approach is Euler integration as specified in Equation (2) with h being the integration stepsize.

$$x_{n+1} = x_n + h \cdot \nu(x_n) \tag{2}$$

This method has first-order accuracy. A more accurate integration method is for example the fourth-order Runge-Kutta method (3)

$$x_{n+1} = x_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5) \tag{3}$$

with

$$\begin{aligned} k_1 &= h \cdot \nu(x_n) \\ k_2 &= h \cdot \nu(x_n + \frac{k_1}{2}) \\ k_3 &= h \cdot \nu(x_n + \frac{k_2}{2}) \\ k_4 &= h \cdot \nu(x_n + k_3) \end{aligned}$$

The approximation can be further improved by adaptively adjusting the integration stepsize h [PFTV88].

In the following terms and concepts often used in the analysis of dynamical systems are shortly discussed.

We start with the nabla operator ∇, which is often used to define other important terms for the analysis of dynamical systems. The nabla operator builds up a vector of the partial derivatives of its operand and is defined as shown in (4) [BS80]

$$\nabla = \left(\frac{\partial}{\partial x_1} \frac{\partial}{\partial x_2} \cdots \right)^T, \quad \nabla f(x) = grad\, f(x), \quad \nabla \nu(x) = J = \frac{\partial \nu}{\partial x} \tag{4}$$

If ∇'s operand $f(x)$ is a scalar function, then $\nabla f(x)$ is the *gradient vector* $grad\, f(x)$ of $f(x)$. If ∇'s operand $\nu(x)$ is a vector function, then $\nabla \nu(x)$ is the *Jacobian* matrix $J = \frac{\partial \nu}{\partial x}$ of $\nu(x)$. Taking the Rössler system $R(x)$ as an example, we obtain its Jacobian matrix by calculating $\nabla R(x)$ as shown in (5) [PJS92]

$$R(x) = \begin{pmatrix} -(x_2 + x_3) \\ x_1 + a x_2 \\ b + (x_1 - c) x_3 \end{pmatrix}, R's Jacobian \nabla R(x) = \begin{pmatrix} 0 & -1 & -1 \\ 1 & a & 0 \\ x_3 & 0 & x_1 - c \end{pmatrix} \tag{5}$$

An often used scalar term is the *divergence* of a flow $div\, \nu(x)$. It can be written as $\nabla \cdot \nu(x)$ or as the trace Tr of ν's Jacobian $\nabla \nu(x)$ (6) [BS80].

$$div\, \nu(x) = \nabla \cdot \nu(x) = Tr(\nabla \nu(x)) = \sum_i \left(\frac{\partial \nu}{\partial x}\right)_{i,i} \tag{6}$$

The divergence basically describes the local amount of outgoing and incoming flow at a specific location in phase space of the dynamical system. If it is zero the amount of incoming flow equals the amount of outgoing flow.

Another important term for the local analysis of dynamical systems is the rotation vector of a flow $rot\,\nu(x)$ [PW94]. This attribute of a flow is often named *vorticity* instead of rotation and is denoted by ω [Han93]. As a third term sometimes *curl* is used instead of rotation. The vorticity/rotation/curl of a flow is defined as given in Equation (7).

$$\omega = rot\,\nu(x) = curl\,\nu(x) = \nabla \times \nu(x) \tag{7}$$

The vector $rot\,\nu(x)$ describes the rotation axis and its length the rotation velocity at position x. Some references define the vorticity slightly different as $\omega = 1/2 \cdot rot\,\nu(x)$. A scalar term related to the vorticity is the *stream vorticity* Ω (8) [SVL91,Han93].

$$\Omega = \frac{\nu \cdot \omega}{|\nu| \cdot |\omega|} = \frac{\nu \cdot (\nabla \times \nu)}{|\nu| \cdot |\nabla \times \nu|}, H_d = \Omega \cdot |\nu| \cdot |\omega| = \nu \cdot \omega \tag{8}$$

Just slightly different from the definition of stream vorticity is the specification of *helicity* or *helicity density* H_d (8) [dLvW93,PvW93]. A helicity density of zero means no stream vorticity. Helicity density increases proportional to the length of ω and ν.

Another term in connection with the rotation of a flow is its *circulation* Γ_C [Laj94]. The circulation of a flow determines if it is possible to use a potential function instead of the vector function ν for analysis purposes. If the circulation Γ_C of a flow is zero for any closed curve C, then a potential function p exists such that $grad\,p(x) = \nu(x)$. In such a case it is often easier to analyse p instead of ν. Additionally the fact $\forall C : \Gamma_C = 0$ implies that there is no rotation at all in the vector field. By using Stoke's equations, Γ_C can be expressed as shown in Equation (9)

$$\Gamma_C = \oint_C \nu(x)\,ds = \int_A rot\,\nu(x)\,dA \tag{9}$$

with A being the surface of an arbitrary volume which contains the closed curve C.

Certain topological structures within phase space are of special interest. Locations $\bar{x}$ with $\dot{\bar{x}} = \nu(\bar{x}) = 0$ are called fixed points, critical points or equilibrium points of the dynamical system. When all trajectories close to an equilibrium point $\bar{x}$ converge to that point, $\bar{x}$ is called an attractor. If all trajectories close to $\bar{x}$ diverge $\bar{x}$ is called a repellor. A trajectory $x(t)$ with $x(t) = x(t+T)\forall t$ is called a periodic trajectory, a cycle or an oscillation. Cycles again can be attracting or repelling. Attracting points or cycles are called *limit sets* of a dynamical system. In addition to limit points and limit cycles other limit sets occur in systems with dimension greater than two. For example in a three-dimensional system a torus can occur as a limit set.

Limit sets of high-dimensional systems may have complex and sometimes even chaotic behavior (e.g., chaotic and strange attractors).

A region within phase space where all trajectories converge to a limit set A is called the *inset* of A. When reversing the flow orientation in a dynamical system (e.g., integration is done backwards in time) the property of attraction and repulsion changes. The inset of a limit set A under reverse integration is called the *outset* of A. The insets of different limit sets are separated by *separatrices* [AS92]. Separatrices segregate regions of phase space with vastly different dynamic behavior.

The topology of a dynamical system can be described by the position and behavior of its limit sets. Analytically the behavior close to a limit set can often be determined by linearizing the dynamical system. Given a dynamical system $\dot{x} = v(x)$ the vector function $v(x)$ can be linearized by a Taylor expansion as follows (10)

$$\dot{(x + \Delta)} = v(x + \Delta) = \sum_{k=0}^{\infty} \frac{1}{k!} (\Delta \cdot \nabla)^k * v|_x \approx v(x) + \nabla v|_x \tag{10}$$

Linearization takes into account only the zero-order and first-order terms in the Taylor expansion. At a fixed point $\bar{x}$ the zero-order term vanishes, i.e., $v(\bar{x}) = 0$. The first-order term corresponds to the Jacobian matrix J. Therefore the evolution of a small perturbation Δ close to a fixed point $\bar{x}$ is guided by the differential equation (11)

$$\dot{\Delta} = J|_{\bar{x}} \cdot \Delta \tag{11}$$

The dynamics close to a hyperbolic fixed point $\bar{x}$ is determined by the eigenvectors e_i and eigenvalues $\lambda_i = a_i + b_i \cdot i$ of the Jacobian J. They are defined as $det(J - \lambda_i \cdot I) = 0$ and $J \cdot e_i = \lambda_i \cdot e_i$. The number of eigenvalues and eigenvectors is equal to the dimension of phase space. If a_i is less than zero convergence and if a_i is greater than zero divergence along the eigenvector e_i occurs. If a b_i is not equal to zero additionally a rotational movement around the fixed point is given. A fixed point is called hyperbolic (i.e., structurally stable and easy to analyse) if no a_i is equal to zero. A fixed point $\bar{x}$ where all eigenvalues are real and negative (positive), is called attracting (repelling) node. In the two-dimensional case a fixed point with one positive and one negative real eigenvalue determines a saddle node.

In case of a discrete dynamical system $x_{n+1} = v(x_n)$ the situation is somewhat similar. Close to a fixed point $x_{n+1} = v(x_n) = x_n$ convergence (divergence) along eigenvector e_i is given if $|\lambda_i| < 1$ ($|\lambda_i| > 1$). Again for b_i not equal to zero rotational movement occurs. Fixed points of discrete dynamical systems are hyperbolic, if $|\lambda_i| \neq 1$ holds for all eigenvalues.

The Jacobian J can also be decomposed into it's symmetric and asymmetric components. Another possibility of analyzing the Jacobian is to transform

J into the local Frenet frame at some point of a trajectory. This allows an easy investigation of the flow behaviour parallel and perpendicular to the flow [dLvW93].

2 Visualization of Dynamical Systems

The visualization of dynamical systems provides insight into the often intricate behavior of such systems [LGWP96]. Many techniques result from the field of experimental fluid flow research (e.g., particle injection or dye advection). Figure 2 shows a classification of visualization techniques with increasing focus from top to bottom. There are techniques to visualize entire classes of dynamical systems (e.g., bifurcation diagrams [AS92]). Systems within a class have different system parameters. Taking a specific set of parameters allows to investigate a single system. The entire phase space may be visualized (e.g., hedge-hog method [PvW93] , spot noise [vW91] , LIC [CL93], topological representation [HH91]). With another focusing operation the visual analysis may be directed to a (small) subset of phase space. Techniques are either direct visualizations of the flow (e.g., LIC) or they show derived quantities, like topological structures. Topological representations show for example limit sets, separatrices, saddle-connections, homo-clinic orbits. These objects are often quite difficult to calculate but suffice to illustrate the qualitative behavior of the flow [HLL97].

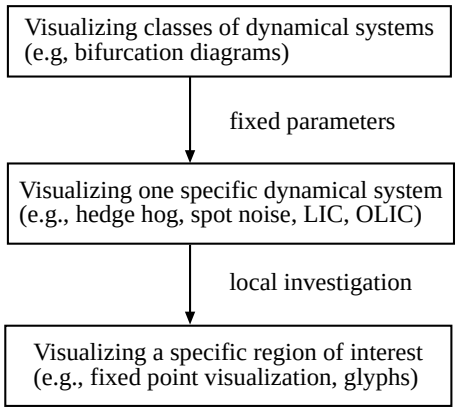


Fig. 2. Visualizing Dynamical Systems (focus increases from top to bottom)

The hedge-hog method displays the tangential directions of the flow at selected positions (e.g., regular grid) of phase space. The length of the vectors encodes flow velocity. Such plots give information about the vector field but do not show important structures, e.g., vortices of the underlying dynamical

system. Although quite feasible for 2D dynamical systems the images for higher dimensions tend to become crowded. Occlusion is a problem in such images which are therefore difficult to analyse.

Often scientists want to investigate a dynamical system at some specific point in phase space. They are interested in local aspects as, for example, velocity, acceleration, and divergence. A vortex or a point near an obstacle may be such a point of special interest. Local properties are often determined by analyzing the Jacobian J at some point of the flow. Visualization techniques were developed that help to investigate these local flow properties. Glyphs are a prominent example for this type of visualization. A glyph is a geometric object whose properties (e.g., length, shape, color) encodes underlying flow properties like acceleration, shear, curvature, torsion, convergance and divergence. In [dLvW93] the glyph is an enhanced three-dimensional arrow which can be positioned interactively or automatically, e.g., along consecutive points of a trajectory.

Streamlines, streaklines and pathlines [Han93] illustrate the temporal evolution of an initial position in phase space. A streamline visualizes a trajectory of a dynamical system. It also describes the path of a single particle in a time-independent flow. A pathline describes the path of a single particle in a time-dependent flow. A streakline visualizes the path of a sequence of particles which are introduced into a time-dependent flow at a fixed spatial position but regularly distributed over time. Timelines on the other hand result by introducing particles at a fixed moment in time but regularly distributed in phase space. A streamline is generated by moving a point (initial position) through the flow. Taking more general objects like lines, circles or implicit surfaces produces streamribbons, streamsurfaces, streamtubes or streamballs [BDH+94].

In [SVL91] an n-sided polygon is positioned perpendicular to the flow and moved along a trajectory to encode local flow attributes, like rotation and shear.

Mathematicians sometimes use Poincaré sections to investigate chaotic and/or strange attractors of dynamical systems [Tso92]. Starting with a continuous dynamical system a Poincaré section defines a discrete dynamical system of lower dimension which is easier to analyse and visualize [LKG97b].

A bifurcation diagram is the most common method for visualizing an entire class of dynamical systems. Such a diagram is constructed by extending the plot of the dynamical system's long-term behavior by additional coordinate axes corresponding to various system parameters. Variations along these axes represent modifications to the model and thus several systems can be illustrated within one image. At certain parameter values the behavior may change qualitatively (bifurcation scenario) therefore these plots are called bifurcation diagrams [AS92].

In the following several recent approaches to visualize dynamical systems are described in more detail. These techniques include: texture based flow

visualization, visualizing high-dimensional dynamical systems, and stream arrows.

3 Texture Based Techniques

Texture based techniques for the visualization of flow fields have been investigated in detail in recent years. Examples are, e.g., [vW91,CL93,MCG94,WG97].

Spot noise is a stochastic texture synthesis technique to visualize scalar fields and vector fields [vW91,dLvW95,dLPV96]. A spot noise texture is constructed by accumulating randomly weighted and positioned spots. Spot noise is a versatile technique where characteristics of the spot are intuitively transfered to characteristics of the spot noise texture. Varying the shape and features of the spot locally enables a local control of the texture. A flow field can be visualized by taking elongated ellipses as spots. The larger axes of these spots are, for example, aligned with the locally varying flow direction. The result is an anisotropic texture which depicts the entire flow field and does not distract the viewer with larger geometric features.

Line Integral Convolution (LIC) smoothes a white noise input texture along (curved) streamline segments [CL93,SH95]. LIC uses one-dimensional filter kernels which are determined by integrating the underlying vector field. The intensity $I(x_0)$ at an arbitrary position x_0 of the output image is calculated by

$$I(x_0) = \int_{s_0-s_l}^{s_0+s_l} k(s-s_0)T(\sigma(s))\,ds, \tag{12}$$

where T is the input texture, $\sigma(s)$ is the parameterized streamline through x_0 ($x_0 = \sigma(s_0)$) and k() describes the convolution kernel. s_l specifies the length of the streamline segment used in the filter operation. The texture values along the streamline segment $\sigma(s)$, $(s_0 - s_l \leq s \leq s_0 + s_l)$, are weighted with the corresponding kernel values $k(s - s_0)$. They are accumulated to give the intensity $I(x_0)$ at position x_0. Various kernel functions $k()$ can be used in the filter operation. For single images a constant filter kernel gives a good impression of the flow direction. Taking periodic low-pass filter kernels and phase shifting these kernels in successive images allows to animate the flow field. The animation shows flowing ripples which also encode the orientation of the flow.

Extensions and performance optimizations of LIC are investigated in [FC95,SH95,KB96,SJM96,IG97,SK97].

The Line Integral Convolution method presented so far does not encode the orientation of a flow within a still image. In Section 3.1 Oriented Line Integral Convolution (OLIC) [WGP97a] is described, which overcomes this disadvantage. Section 3.2 discusses a new technique for Fast Rendering of

OLIC images (FROLIC) [WG97]. OLIC, FROLIC, and some other visualization techniques have been implemented within a Java applet. The applet can be accessed at `http://www.cg.tuwien.ac.at/research/vis/dynsys/frolic/`.

3.1 Oriented Line Integral Convolution (OLIC)

LIC images encode flow direction and velocity magnitude, but they do not show the orientation of the flow in still images. Flow orientation can be illustrated through animation. But there are cases where only still images are available or necessary, e.g., reproduction of vector fields in books or journals.

Furthermore LIC images are characterized by high spatial frequencies normal to the flow. This gives a good impression of the overall vector field, but is susceptible to aliasing problems in case an image has to be manipulated like, e.g., resized or printed. Oriented Line Integral convolution (OLIC) [WGP97a] was designed to show the orientation of a flow even in still images and it is not as much prone to aliasing effects as LIC. There are two major differences between LIC and OLIC. LIC images typically use dense noise textures wheras OLIC utilizes only sparse textures. A sparse texture can be thought of as a set of ink droplets which are thinly distributed on a sheet of paper. The vector field smears these ink droplets but the ink droplets are so far apart from each other that blurred traces of droplets usually do not overlap. The second difference between LIC and OLIC is that OLIC uses asymmetric convolution kernels (figure 3). A ramp-like kernel as in figure 3 produces traces of droplets with intensity varying along the streamline. As a sparse texture is taken traces do not overlap very much and the orientation of the flow is visible in still images.

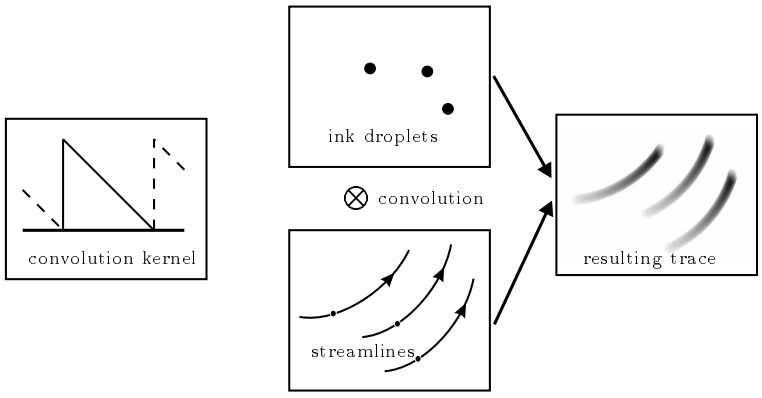


Fig. 3. OLIC: LIC with sparse texture and ramp-like kernel-function

In figure 4 the difference between LIC and OLIC is clearly visible. Figure 4(a) shows the LIC image of a circular flow. In this image it is not recognizable if the flow is in clockwise or counterclockwise orientation. Figure 4(b) shows the OLIC image of a circular clockwise flow and figure 4(c) shows the OLIC image of a circular counterclockwise flow. The additional information in the OLIC image is gained at the expense of spatial resolution.

(a) (b) (c)

Fig. 4. LIC image of circular flow (a), OLIC image with clockwise flow (b), OLIC image with counterclockwise flow (c)

The initial positions of the droplets in the sparse texture must be selected carefully to avoid the formation of undesirable patterns in the OLIC image. This topic is dealt with in more detail in section 3.3. OLIC allows to encode flow velocity by the length of the traces of individual droplets.

3.2 Fast Rendering of OLIC (FROLIC)

One characteristic feature of OLIC is the usage of sparse textures. Convolution of a sparse texture is not as difficult as convolution with a dense texture. This can be used for Fast Rendering of OLIC images (FROLIC) [WG97].

FROLIC calculates an approximate solution to the exact convolution result of OLIC thereby achieving a considerable speed-up. With a sparse texture the convolution at a specific point of the result image involves at most one single droplet. Each droplet produces a trace with intensity increasing from tail to head. Due to the circular shape of a droplet the intensity varies slightly along the breadth of a trace as well (figure 5(a)). As a trace is rather small FROLIC approximates the shape of a trace by a set of small, possibly overlapping disks (figure 5(b)). The disks are positioned along a short portion of a streamline. Each disk has constant intensity, but the intensity varies between adjacent disks. If n disks are taken to approximate a trace then intensity increases in n discrete steps from tail to head of the trace. The intensity of adjacent disks is increasing to simulate the continuous ramp kernel of the OLIC method. Although being an approximative variant of a

(a) (b)

Fig. 5. Exact trace of a droplet with OLIC (a) and approximated trace of a droplet with FROLIC (b)

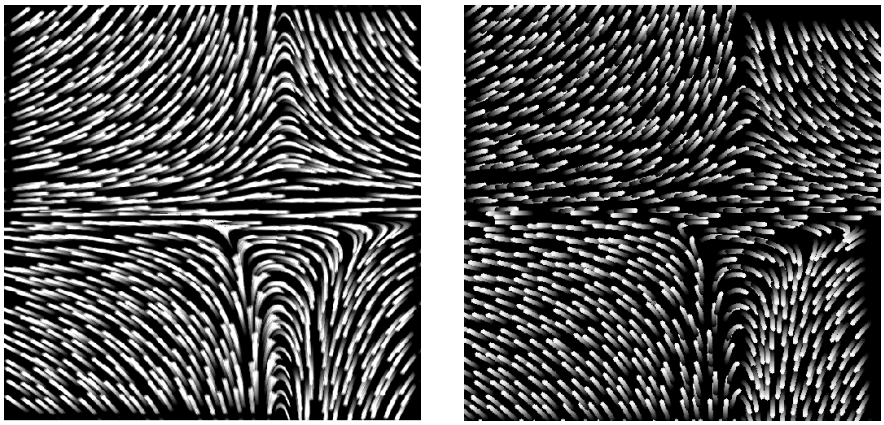
LIC-type algorithm, FROLIC is also somewhat in the spirit of iconic vector field representations. Such approaches are, e.g., spot noise [vW91], surface particles [vW93a], and particle traces on 2D surfaces [MCG94].

The FROLIC calculation is done as follows: for each droplet a short streamline portion is calculated by integrating the underlying flow field. The length of a streamlet indicates local flow velocity. A prefixed number of disks is positioned in regular intervals along a streamlet. The processing order is from tail to head of the trace, i.e., darker disks are drawn first and might be partially occluded by brighter disks which are drawn later on.

The main advantage of FROLIC as compared to OLIC is that drawing disks is much faster than doing a costly convolution calculation. Instead of calculating the result image pixel per pixel as OLIC does, only the rather small set of droplets has to be processed. In an image with resolution of 600x600 about 1000 droplets are sufficient. Furthermore drawing simple geometric primitives like disks can be done with hardware support. During experiments we found that the approximation error introduced by the FROLIC method is well justified by the efficiency gain. Investigations show that FROLIC (without hardware supported rendering) is approximately two orders of magnitude faster than OLIC. Figure 6 gives a comparison between an OLIC and a FROLIC image. A thorough comparison between OLIC and FROLIC is given in [WG97].

3.3 Droplet Texture Design

This section deals with the placement of droplets on a sparse input texture. There are two criteria which should be optimized but which are opposed to each other. One criterion calls for a dense filling of the output image with streamlets. This ensures that most of the vector field information is represented in the output image. The second criterion is that overlapping streamlets should be avoided as far as possible in order to clearly illustrate flow orientation.



(a) (b)

Fig. 6. Econometric model with OLIC (a) and FROLIC (b)

Finding an optimal droplet distribution in the input texture so that a tight packing of streamlets results in the output image is quite intricate. The optimal texture depends on the underlying vector field as well as on the chosen minimal and maximal streamlet lengths. Furthermore changing the position of a droplet has nontrivial consequences concerning the induced streamlet. The streamlet may change its length or shape. This complicates filling algorithms which are based on distributing and moving droplets in the input texture. Another consideration is that the arrangement of the streamlets should not produce macro structures which are easily perveived by the human visual system and which disturb the interpretation of the flow data. Such macro structures might result for example if streamlets are exactly aligned along a specific streamline or the alignment of streamlets is such that they form wavefront patterns.

There has already been work on optimal streamline placement [JL97], [TB96]. Turk and Banks [TB96] use an energy function to guide the placement of streamlines. The resulting images look somewhat like elegant hand-designed streamline drawings but the optimization process itself is quite costly.

Our task deals only with the placement of short streamlets and is therefore inherently simpler than the streamline placement in [TB96]. Simple approaches for droplet placement are random distribution and placement of droplets on a regular or jittered grid. If the distance of adjacent grid points is in the same order as the maximal streamlet length then overlapping may occur but is usually not a severe problem. If overlapping of streamlets shall be avoided entirely, a distance image is used which has the same resolution as the final output image. For each pixel the distance image contains the

distance to the closest streamlet drawn so far. A new streamlet candidate is calculated and if it is too close to a previously drawn streamlet it is discarded. Otherwise the streamlet is drawn into the output image as a set of disks and the distance image is updated. For further details see [WG97]

3.4 Animation of OLIC and FROLIC

The animation of OLIC images can be achieved by simply phase shifting the convolution kernel for each frame of an animation sequence. The phase shift is adapted to the length of the trace of a droplet. Short traces have small phase shifts and long traces have large phase shifts. Initially each droplet is assigned a random phase shift (offset) to avoid synchronization artefacts.

This approach can also be adapted to FROLIC. With FROLIC a streamlet consists of a set of disks with varying intensity. These intensities are cycled to convey to the viewer the impression of motion. The spatial position of streamlets is not changed. In the following we will discuss two algorithms how to realize animation of FROLIC images. The first algorithm is based on the fact that a linearly increasing intensity function has to be cycled. Each streamlet is again assigned a random initial phase shift to avoid synchronization effects. For each streamlet the current position c indicates the disk with highest intensity. Given frame i of the animation sequence the following frame $i+1$ is constructed by reducing the intensity of all pixels of frame i by a fixed amount. This amount is equal to the intensity difference between adjacent disks. Frame $i+1$ is built from the intensity-reduced frame i by drawing a single disk for each streamlet. These disks have highest intensity and are positioned at location $(c+1) mod\, n$. n is the number of disks used to represent a streamlet. Both operations (i.e., intensity reduction and disk drawing) together cycle the intensity ramp one disk along the flow. Both operations are easily and efficiently realizable.

The human visual system is very sensitive to appearing or disappearing bright spots. This can be a problem for cycling an intensity ramp along a fixed streamlet, as a bright disk disappears at the head of a streamlet and reappears at the tail of the streamlet. The impression of a pulsating effect can be, however, avoided by using a filter (e.g., Gaussian) which attenuates the intensity at the beginning and the end of a streamlet.

Color-table animation is the second approach for efficiently animating FROLIC images. With a color table the intensity value of a pixel is specified indirectly. Each pixel is assigned a color-table index which points to a specific entry in the color table. Available intensities or colors are stored in the color table itself. Color-table animation only changes the entries of the color table instead of changing the image.

The color table holds the intensity ramp, i.e., successive color-table entries contain increasing intensity values. Animation is achieved by cycling the intensity values in the color table itself. The random initial phase shift (offset)

is realized by starting each streamlet with a randomly selected color-table index.

3.5 Virtual Ink Droplets

Virtual Ink Droplets [LKG97a] are based on the physical model of smearing ink over a sheet of paper. Virtual Ink Droplets produce images similar to OLIC and FROLIC. The simulation of physical parameters, e.g., pager roughness, allows to encode additional flow parameters. A sparse input texture similar to the one used for OLIC is interpreted as an ink-concentration field over a sheet of paper. Ink is advected according the underlying flow field. Spatially varying absorption properties of the paper are taken into account during the advection process. The roughness of the paper may, for example, be used to emphasize interesting areas of the flow. Ink advection is calculated incrementally. During one step part of the ink is absorbed while another part is advected along the flow. Figure 7 shows an example of using Virtual Ink Droplets. A squared paper was used as background image to highlight the metaphor of smearing ink over paper. In this case paper roughness was modulated to depict the coordinate axes and labels. This can be interpreted as writing text with a wax pencil on a squared-paper background. Similar techniques, i.e., advection of concentration fields, have also been studied in the computational fluid dynamics literature [vW93b].

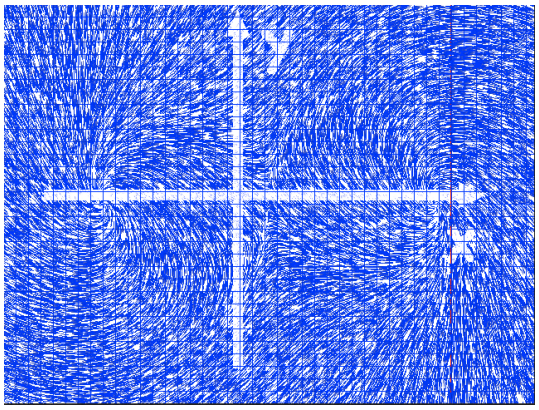


Fig. 7. A dynamical system with three fixed points visualized with virtual ink droplets

4 Visualizing High-Dimensional Dynamical Systems

In recent years scientific visualization has been driven by the need to visualize high-dimensional data sets within high-dimensional spaces. However most visualization methods are designed to visualize point sets. Typically these methods show statistical features like correlations, clustering or outliers. This section on the other hand deals with the visualization of trajectories of high-dimensional dynamical systems.

Several visualization methods for high-dimensional data can be distinguished. The visualization of high-dimensional data often uses a combination of two or more of these methods, e.g., color coding is used on focused data and assisted by interactive sonification. Interactive manipulation and exploration introduces time as a fourth dimension into the space where the data is explored.

Attribute mapping is one of the most common methods to visualize high-dimensional data. This method uses one or two-dimensional lattices to define some simple geometric primitives, e.g., contours or planes. The attributes of these geometric primitives can be used to visualize the remaining variables. The most often used attribute is the color of the geometric primitive. Color coding can be employed to display up to three variables. Each of these variables is mapped to one component of the underlying color model. The most common color models are the RGB model and the HLS model. A major advantage of color coding is the fact that it is very often used (e.g., weather forecast maps). Therefore many users are familiar with this kind of visualization. Another advantage is the easy calculation and interpretation of color coded images. A disadvantage is that colors do not have a unique order, so color coded images have to show a color legend to allow an exact interpretation. Another disadvanatage is the perceptually non-uniformness of the RGB model and HLS model.

Geometric coding is used for displaying high-dimensional data on a low-dimensional lattice by displaying distinct geometric objects (e.g., glyphs or icons) within the lattice and mapping the high-dimensional data to some geometric features or attributes of these objects. Chernoff Faces are an early example of geometric coding [Che73]. A glyph is a generic term describing a graphical entity whose shape or appearance is modified by mapping data values to some of its graphical attributes. An interactively positioned glyph adapts its appearance according to the underlying data. Variables can be mapped to the length, shape, angle, color and transparency of the glyph. Examples of this kind of visualization are given in [Ker90,dLvW93,PvWPS95]. An icon is a generalization of a single pixel to higher dimensions having multiple perceivable features and attributes. The fact that shape and color are perceptually separable features is used for the display of color icons. They merge separable features by using color, shape and texture perception to code multiple variables. In [Lev91] an icon is presented that allows to encode six different parameters by color coding six different lines within a square icon.

Individual variables are not recognizable any more, but correlation patterns appear.

Sonification is another method of making more than three dimensions accessible to a researcher. A sound is produced according to the mapped parameters [GS90]. Variables can be mapped to the loudness, the pitch and even to the orchestration of the sound. One disadvantage of these methods is that the various parameters that characterize a sound influence each other, i.e., a sound at a constant volume but with changing tune is perceived as if the volume changes too. Nevertheless sonification is a good tool for visualizing high-dimensional data sets because it stimulates a different sense organ and thus may avoid overloading the visual system.

Another obvious way of visualizing high-dimensional data sets is the reduction of dimension. This can be done by either focusing, where only part of the whole data set is shown, or by linking, where some focused parts are linked together to represent the whole data set. Focusing techniques may involve selecting subsets, reduction of dimension by projection, or some more general manipulation of the layout of information on the screen. Examples for subset selection techniques are panning, zooming, and slicing. Reduction of dimension can be achieved by simply projecting high-dimensional spaces along some axes into a low-dimensional space and/or color coding of multi-parameter images. Techniques for more general layout manipulation include a variety of techniques for adapting to a user's point of interest such a fisheye views [Fur86] and rooms [HC86]. One consequence of focusing is that each view will only convey partial information about the data. This can be compensated by linking several focused visualizations. Linking can be done by sequencing several visualizations over time (guided tour) or by showing them in parallel simultaneously. The parallel visualization can be done in separate windows as for example with the well known scattered data plots. It can also be done within one single image by using parallel coordinates [ID90], dimensional stacking [LWW90], or hierarchical axis [MTS91].

Parallel coordinates [ID90] represent dimensions on parallel axes. Each axis represents one coordinate component. All axes are arranged orthogonal to a horizontal line uniformly spaced on the display. An n-dimensional point of the data set is displayed as a polyline that intersects the parallel coordinate axes at the corresponding coordinate values of the data point. This method allows the detection of special characteristics of the data by looking at the patterns that are produced by the polylines. If, for example, all data points are colinear in n-space, then all polylines will intersect each other at specific points between the (vertical) parallel coordinate axes. Thus a line of n-space can be visualized by a set of points between the parallel coordinate axes. This gives a duality between points and lines which is an interesting feature of parallel coordinates. By interactively brushing through the data set statistical characteristics like outliers and clusters can be recognized easily.

Displaying an n-dimensional trajectory is an important task to allow a direct global visualization of the behavior of a dynamical system. One possi-

ble way for doing this uses the fact that an n-dimensional directional vector can be described by $n-1$ angles. Since the direction changes smoothly along a trajectory these angles can also be used to describe the behavior of a trajectory thus reducing the dimension by one. This would allow us to display four-dimensional trajectories in three-space. The investigation of topological structures such as two-dimensional and three-dimensional manifolds in four-space has already been done by Hanson [HC93]. In the following more general methods for visualizing n-dimensional trajectories with $n \geq 4$ are discussed [WLG97].

4.1 Extruded Parallel Coordinates

Extruded parallel coordinates are based on parallel coordinates. With parallel coordinates a trajectory is sampled during its evolution at discrete points in time $\{x(t_0), x(t_1), x(t_2), \ldots\}$ and its coordinates are inserted as polylines in a parallel coordinate system (see left side of Figure 8). Instead of using the same coordinate system for each sample we now move the parallel coordinate system along the third spatial axis. The polylines of the samples can be viewed as cross sections of a moving plane with a complex surface which defines the trajectory. The right side of Figure 8 shows this surface and the moving parallel coordinate system at the end of the surface.

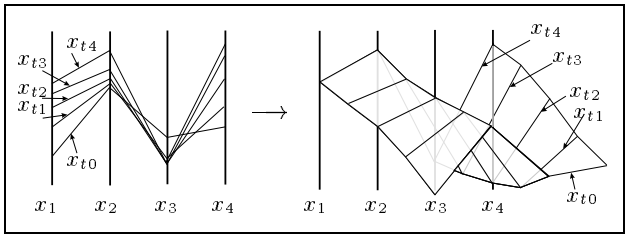


Fig. 8. A discrete sampled trajectory in parallel coordinates (left) and a three-dimensional extruded surface defining the same trajectory (right)

The geometry of the surface can be generated and modified fast and easily allowing an interactive exploration of trajectories in the dynamical system. All exploration methods used with parallel coordinates can be used as well since rotating the surface and parallel projecting it reveals exactly the parallel coordinate representation. This can be used as a starting point for the exploration of the trajectory. Clustering and correlation can be visually detected [ID90]. Rotating the surface a little bit reveals the evolution of the trajectory over time without any animation methods that would have to be used for parallel coordinates.

Convergence or divergence can be observed by varying the starting coordinates of the trajectory slightly. The changing shape of the surface shows for

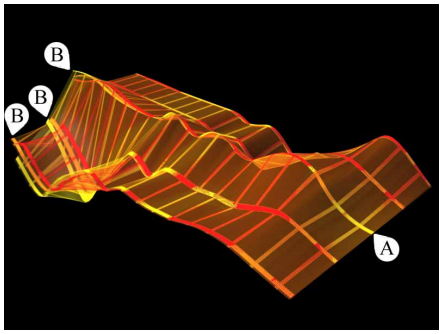


Fig. 9. Small jitter at the starting point (A) of a chaotic attractor yields large effects at certain coordinate components (B) after some period of time

each dimension if its attracted or repelled by some topological structure. If for one dimension a whole interval is used for the starting points of trajectories the surface expands to a volume which again can be used for a structural analysis of the underlying dynamical system.

In Figure 9 extruded parallel coordinates illustrate a trajectory of a chaotic attractor (five-dimensional system). To show the chaotic behavior of the attractor, the starting point of the third dimension is slightly jittered (point A) and three different trajectories (shown with different colors) are superimposed. The tiny differences of the starting coordinates produce large changes after a few integration steps (points B). Interestingly these differences are noticeable only in the first three dimensions so far (integration over a longer time interval shows diverging behavior in the fourth and fifth dimension also).

4.2 Linking with Wings

Linking with Wings is a new method of linking data. Two arbitrary dimensions of the high-dimensional system are selected and displayed as a two-dimensional trajectory within a base plane (the high-dimensional trajectory is projected into a two-dimensional subspace). The third dimension (along the z-axis) can now be used to display a third variable over the base trajectory. If the resulting three-dimensional trajectory is connected with the base trajectory this connection can be thought of as a wing on the base trajectory. This wing can be tilted at each point within a plane normal to the base trajectory. When different tilting angles are used several additional dimensions can be linked to the base trajectory on separate wings (see Figure 10).

Theoretically any number of wings can be added to display high-dimensional trajectories. As the number of wings increases occlusion might become a severe problem. To avoid this the wings can be rendered transparently with opaque tubes at the top. Again this method is easy to implement and fast allowing its use within an interactive exploration tool. Such a system could

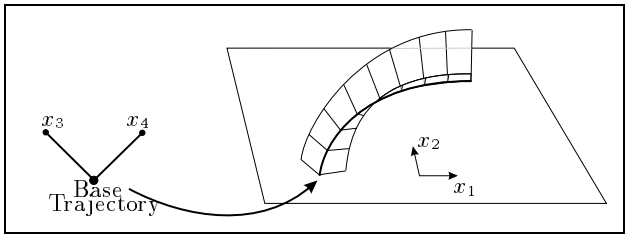


Fig. 10. Two-dimensional base trajectory with two wings (for the third and fourth dimension) linked to it

for example allow to animate a flight along the base trajectory, where the change in the variables linked to wings can be seen easily.

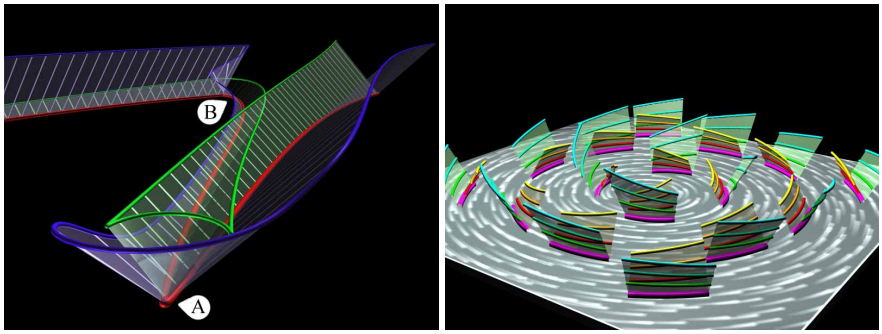
The wings can also be textured with a grid texture allowing an exact measurement of the wing dimensions. To overcome the problem of occlusion texture can be used to modulate the transparency of the wings. Negative values of variables displayed on wings can be shown by expanding the wing to the opposite side of the base trajectory or by using an additive offset for each wing so that the minimum of each variable is mapped to the base trajectory. For this approach the zero line has to be encoded on the wing. This can again be done by using a specific texture.

When a four-dimensional trajectory has to be displayed, the angles of the wings can be chosen to be $\frac{\pi}{2}$ and $-\frac{\pi}{2}$. In this case wings lie within the base plane. The trajectory is shown in a two-dimensional image without any projectional distortions .

Self intersection of the wings can be a problem. The size of the wings should be chosen to be rather small with respect to the size of the base trajectories. This is required to avoid massive occlusion. Furthermore the angles of the wings must not be too big. This ensures that the occurrance of self intersections of the wings within regions where the base trajectory exhibits big curvature is not a severe problem.

In Figure 11(a) wings are applied to a base trajectory of a four-dimensional econometric model [WGP97b], which describes the interactions of population, economy, environmental health, and pollution. Rather big wings with larger angles are used intentionally to show the artifacts due to intersecting wings at the cusp of the base trajectory (point A). In spite of the self intersections of the wings the overall behavior of the system is visible: the green trajectory declines constantly along the whole trajectory, whereas the blue trajectory (describing the environmental health) collapses at point A and regenerates at point B (here the wings intersect again).

Figure 11(b) shows a hedge-hog visualization of a four-dimensional data set. On a regular four-dimensional grid flow directions are depicted. The visualization shows a cyclic behavior in the first and second dimension, whereas the third dimension is attracted and the fourth is repelled by the origin. The cyclic behavior in the ground plane is additionally visualized by using OLIC.



(a) (b)

Fig. 11. Self intersecting wings due to high curvature of the base trajectory (a) – Hedge-hog visualization of a four-dimensional data set (b)

4.3 Three-dimensional Parallel Coordinates

Three-dimensional parallel coordinates are again based on the parallel coordinate method. The basic idea of parallel coordinates is to depict each coordinate component on a one-dimensional space. All these one-dimensional spaces are put together within a two-dimensional space and linked with one-dimensional polylines. All information is packed in the two-dimensional space. Since the visualization of three-dimensional structures poses no problem we increased each dimension of the parallel coordinate method. The basic information now resides in separate two-dimensional spaces (planes) where two dimensional projections of trajectories are shown. These planes are combined within three-space and linked by surfaces which connect the separated projections of trajectories (see left part of figure 12).

The positioning of the planes is more flexible in comparison to the parallel lines of the parallel coordinate method. The planes can be moved and rotated within three-space to avoid occlusion in different regions of the structure. The right side of Figure 12 shows two coinciding planes, where the connecting surface extends into the third dimension to give a better overview of the linking.

Placing the planes orthogonal to each other as shown on the left side of Figure 13 is another possibility of showing the structure of the linkage of the separated trajectories. All these arrangements can be stacked to allow the representation of even more dimensions. An example is given in the right side of Figure 13, where an eight-dimensional trajectory is shown. Since the planes and linking surfaces are rendered transparently or are approximated with lines, no massive occlusion occurs (notice that all four separate projections of the trajectory can be seen easily).

If the structure of the trajectory is more complex, as for example in the case of a trajectory of a chaotic attractor, the resulting visualization can be

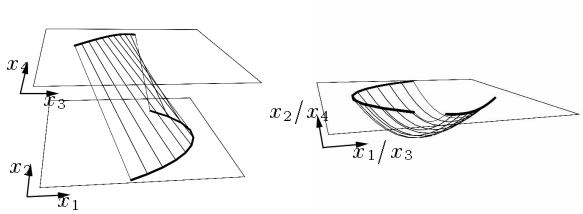


Fig. 12. Linking parallel planes instead of lines extends the idea of parallel coordinates by one dimension (left); these parallel planes can coincide, with three-dimensional linking surfaces (right)

rather crowded. Since the rendering of the presented structures is fast, interactive brushing methods can be used to overcome this problem. For instance the linking surface can be rendered transparently, and only a small temporal interval of the linkage is rendered opaquely. When the interval is moved corresponding parts of the trajectories can then be detected (see Figure 14(b)).

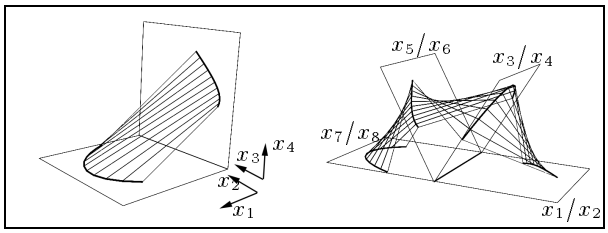
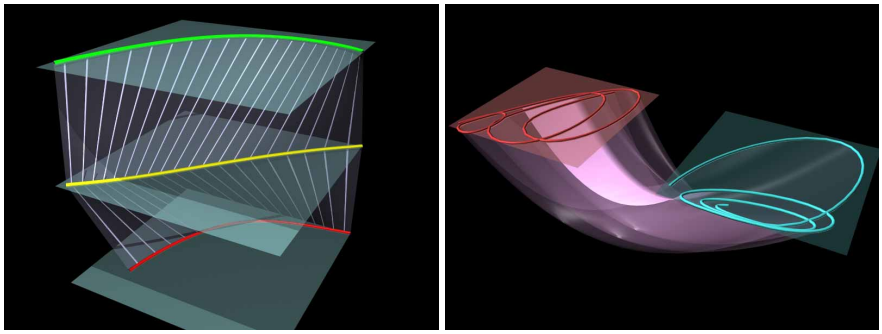


Fig. 13. Planes might be orthogonal (left) and stacking allows an arbitrary number of dimensions to be displayed (right)

When within one dimension a whole interval is chosen as initial region, the linking surfaces expand to linking volumes, whose changing thickness reveals convergent and divergent regions of the trajectory. Again this interval can be chosen interactively for each dimension allowing a quick exploration of the behavior of the dynamical system.

In Figure 14(a) a six-dimensional predator-prey system is stacked with the extended parallel coordinate method. Due to the simple shape of the separated trajectories, the linking surfaces can be seen easily and so the whole dynamics of that specific trajectory is visible. Figure 14(b) on the other hand shows a trajectory of a complex dynamical system derived from a Lorenz system. Here the structure of the linkage can not be perceived easily. Therefore a small temporal interval has been highlighted on the linking surface. This interval can be animated or interactively moved forward and backward to reveal the structure of the linkage.



(a) (b)

Fig. 14. A six-dimensional stacked predator-prey model with simple linkage (a) – Complex linkage with highlighted time interval for the trajectory of a chaotic attractor (b)

Java applets concerning the above techniques have been implemented and can be accessed at `http://www.cg.tuwien.ac.at/research/vis/dynsys/NDim97/`.

5 Advanced Streamsurface Representations

Streamsurfaces are generated by following a line of initial positions through the phase space of a dynamical system. They help to understand topological structures in phase space. The occlusion problem may, however, become severe whenever large scale opaque surfaces are used in the visualization. An interesting technique to deal with the problem of large scale occlusion is utilized in [AS92]. The authors discuss various types of three-dimensional dynamical systems by using hand-drawn illustrations representing the topological structure of the systems. Streamsurfaces make up an important part of most of their images. To reduce the negative effects caused by occlusion they use only arrow-shaped parts of a streamsurface instead of the whole surface. Additionally they use arrow-shaped holes within streamsurfaces to diminish occlusion and include directional information in the images. They also use simple textures in their hand-drawn illustrations to convey a better understanding of the shape of objects in phase space. Highly occluding streamsurfaces occur for example in dynamical systems that exhibit mixed-mode oscillations [MSLG97]. A typical streamsurface for such a system occludes important parts of itself by forming a shell-like roll with several turns. Various extensions of streamsurface representation are discussed in the following sections to cope with this problem.

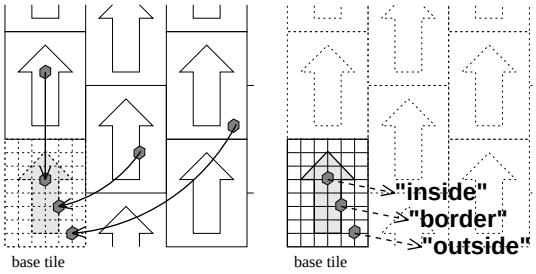


Fig. 15. Mapping of streamsurface vertices into the base tile and classification.

5.1 Stream Arrows - Segmenting Streamsurfaces

With higly occluding streamsurfaces an opaque representation of the surfaces is not appropriate. It is also not a good idea to represent a streamsurface being homogeneously transparent. Portions of the image may be covered by many layers of semi-transparent parts of the streamsurface. This makes it quite difficult to recognize the spatial arrangement of the objects. Several methods modulate the transparency locally within a surface [IFP96,Rhe96].

Stream arrows [LMGP97] also belong in this area of research. Stream arrows are a computer based realization of the ideas of [AS92] by automatically segmenting a streamsurface with arrows. Semi-transparent parts of the streamsurface allow the viewer to see through and perceive parts of the model that otherwise would have been occluded. The opaque part of the streamsurface on the other hand still conveys a clear impression of the shape of the streamsurface. See Figure 17 for a typical image. The arrows used in the segmentation are distorted according to the local flow and also indicate the flow direction. Long stream arrows, for example, visualize a streamsurface region of relatively high velocity. Comparing the width of an arrow's head to the width of it's tail indicates regions of convergent or divergent behavior. Stream arrows can be animated by moving them along streamlines. Only a short animation sequence has to be calculated which is then cycled.

Stream arrows are generated by one of two approaches. On the one hand they can be realized as an alpha-texture that is composed of a set of regularily arranged arrows. Either the stream arrows or the remaining surface portions can be rendered transparently. With this approach the geometric database of the streamsurface which is a set of triangles is not modified.

On the other hand stream arrows can be represented directly by geometrically separating the original streamsurface into a set of arrow-shaped patches and the remaining surface portions which contain holes. This approach allows to further operate on both parts independantly from each other. The streamsurface is tiled with a base stream-arrows tile (texture) which contains a single arrow. The shape of the arrow, its size, and the tesselating scheme are given as parameters of the base tile (see Figure 15).

Each triangle of the streamsurface is classified with respect to the base tile as being inside, outside or on the border of a stream arrow. Triangles lying on the border of a stream arrow are segmented into an inside and an outside portion.

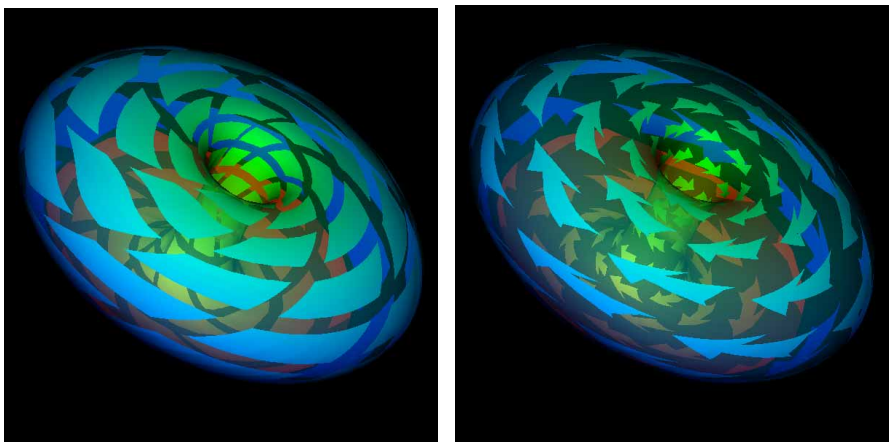


Fig. 16. Using different shapes with the stream-arrows technique

Although the stream-arrows technique is based on the idea of mapping arrow-shaped objects to streamsurfaces, the same method can be easily used to map arbitrary shapes to streamsurfaces. Figure 16 shows a streamsurface computed for a flow around a 3D torus. The streamsurface is segmented by using two different base tiles.

Figure 17 illustrates the stream-arrows texture applied to a streamsurface typical for mixed-mode oscillations. In this case a streamsurface is generated by following a line of initial conditions (colinear to the c-axis) through phase space. As the streamsurface evolves over time it forms a shell-like roll. Color coding was used to distinguish states later in time from earlier ones, i.e., as time goes on the corresponding streamsurface regions are colored blue, green, yellow, and red, respectively. Due to local varying velocities and large divergence, arrows located in the red part of the streamsurface are bigger than those in the green region.

5.2 Hierarchical Stream Arrows

The stream-arrows approach is based on a regular tiling of texture space. This is not well-suited for streamsurfaces that spread over regions of high divergence or convergence. The arrows become either too big or too small in certain areas. To eliminate this undesirable effect a hierarchical extension to the stream-arrows technique is used [LMG97].

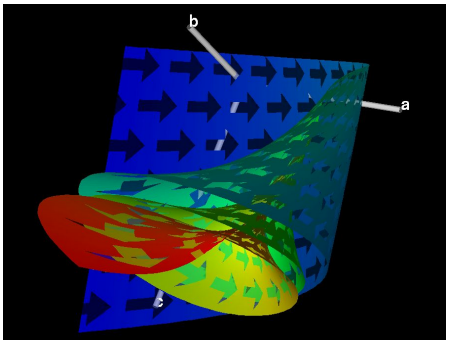


Fig. 17. Streamsurface with stream arrows

The goal is to generate stream arrows which are almost equal-sized in the rendered image. When the flow, for example, diverges locally, the method switches discretely to the next detailed level of stream arrows.

The *hierarchical stream-arrows texture* consists of a stack of stream-arrows textures, where successive levels contain arrows with decreasing size. The hierarchical stream-arrows texture is specified by the shape of one base tile, i.e., the outline of an arrow, and two vectors dc and dr (see Figure 18) which define the offsets between adjacent columns and rows of arrows, respectively.

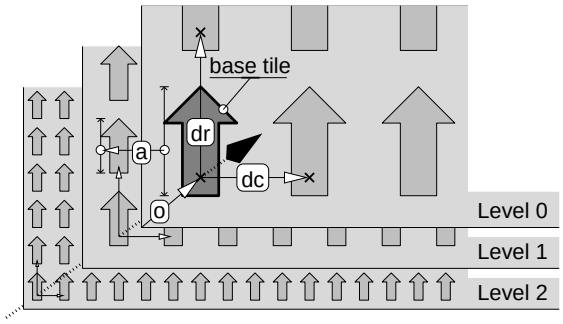


Fig. 18. Hierarchical stream arrows texture, specification parameters

Additionally there is a factor a which represents the scale relation between level i and $i+1$. If a=1/2, for example, the size of stream arrows is doubled, when the algorithm switches to the next coarser level. Finally there is a vector o, which is the offset of the entire texture with respect to the origin of texture space. Offset vector o becomes important, when animation is applied. Due to this specification each stream arrow can be addressed by exactly one ID given by three numbers. ID ($level$, col, row) identifies one stream arrow as a

copy of the base tile, first translated by $\mathsf{o}+(\mathsf{dc}\cdot col, \mathsf{dr}\cdot row)$, and then scaled about the origin by a^{level}.

The algorithm segments the streamsurface triangle by triangle. For each triangle the corresponding `level` in the hierarchical stream-arrows texture is determined by comparing the size of the triangle in texture space to its size in phase space coordinates. This ratio is used to find the most appropriate level in the stack of stream-arrows textures. Then all tiles of the chosen level, which might intersect the triangle, are determined. It must be assured that neighboring triangles are consistently segmented with arrows of the same level in the hierarchical stream-arrows texture. For details see [LMG97]. Figure 19 shows two examples of dynamical systems with largly varying divergence where hierarchical stream arrows are useful.

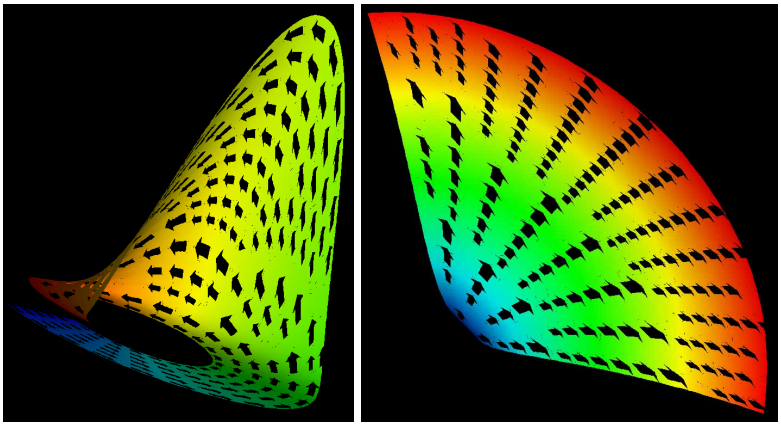


Fig. 19. Hierarchical stream arrows, two examples

5.3 Anisotropic Spot Noise as Streamsurface Texture

Spot noise [vW91] can be efficiently used to emphasize streamlines and timelines simultaneously within a streamsurface. An anisotropic spot, e.g., cross or hash, accentuates horizontal and vertical directions in the resulting spot noise (see Figure 20).

The anisotropic spot noise of Figure 20 is mapped onto the streamsurface so that the horizontal direction of the texture is aligned with the streamlines and the vertical direction of the texture is aligned with the timelines. Figure 21(a) shows a textured streamsurface of the mixed-mode oscillations model. The upper part of the streamsurface is rendered semi-transparently. In Figure 21(b) the separated stream arrows are slightly shifted in the direction perpendicular to the remaining streamsurface portions.

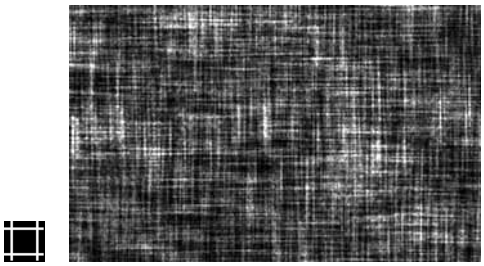
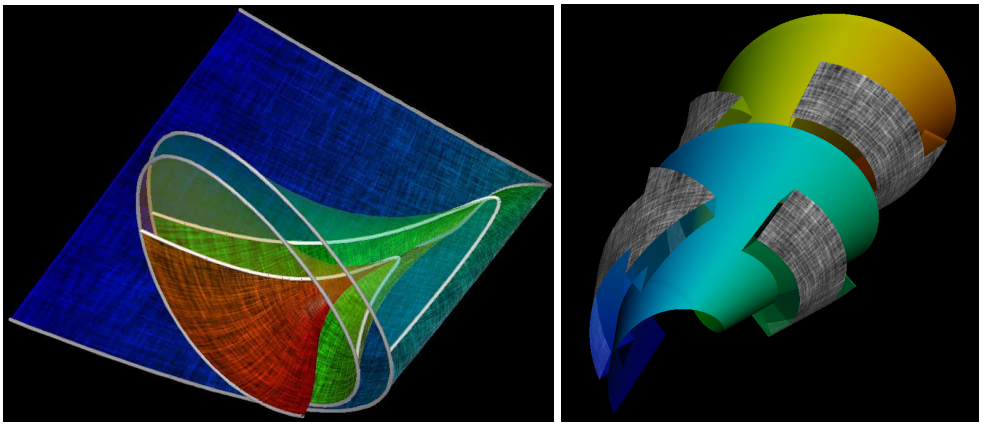


Fig. 20. spot (enlarged) and the resulting spot noise texture



(a) (b)

Fig. 21. Streamsurface with anisotropic spot noise texture (a) –Stream arrows shifted out of the stream surface and anisotropic spot noise (b)

6 Conclusion

Graphical analysis is a valuable tool in the investigation of analytically defined dynamical systems. This paper discusses some theoretical concepts concerning such systems. In addition to a direct representation these concepts allow the display of derived characteristics of dynamical systems. An overview of various visualization techniques shows that different investigation goals lead to greatly varying graphical representations. Three major research directions, i.e., texture based techniques, visualizing high-dimensional dynamical systems, and advanced streamsurface representations, emphasize the diversity of graphical tools available for the analysis of analytically defined dynamical systems.

7 Acknowledgements

The authors would like to thank Andreas König, Lukas Mroz, Werner Purgathofer, Zsolt Szalavári.

References

[AP90] D. K. Arrowsmith and C. A. Place. *An Introduction to Dynamical Systems*. Cambridge University Press, 1990.

[AS92] R. H. Abraham and C. D. Shaw. *Dynamics - the Geometry of Behavior*. Addison-Wesley, 1992.

[BDH+94] M. Brill, W. Djatschin, M. Hagen, S. V. Klimenko, and H.-C. Rodrian. Streamball techniques for flow visualization. In *Proceedings Visualization '94*, pages 225–231, October 1994.

[BS80] I. N. Bronstein and K. A. Semendjajew. *Taschenbuch der Mathematik*. Harri Deutsch, Zürich, 1980.

[Che73] H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 68:361–368, 1973.

[CL93] B. Cabral and L. C. Leedom. Imaging vector fields using line integral convolution. In *Proceedings SIGGRAPH '93*, pages 263–272, 1993.

[dLPV96] C. W. de Leeuw, F. H. Post, and R. W. Vaatstra. Visualization of turbulent flow by spot noise. In *Virtual Environments and Scientific Visualization'96*, pages 287–295. Springer, 1996.

[dLvW93] W. C. de Leeuw and J. J. van Wijk. A probe for local flow visualization. In *IEEE Visualization '93 Proceedings*, pages 39–45. IEEE Computer Society, October 1993.

[dLvW95] C. W. de Leeuw and J. J. van Wijk. Enhanced spot noise for vector field visualization. In *IEEE Visualization '95 Proceedings*, pages 233–239. IEEE Computer Society, October 1995.

[FC95] L. K. Forssell and S. D. Cohen. Using line integral convolution for flow visualization: Curvilinear grids, variable-speed animation, and unsteady flows. *IEEE Transaction on Visualization and Computer Graphics*, 1(2):133–141, June 1995.

[Fur86] G. W. Furnas. Generalized fisheye views. In *Proceedings of CHI '86, Human Factors in Computing Systems*, 1986.

[GS90] G. Grienstein and S. Smith. The perceptualization of scientific data. In *Proceedings of SPIE '90*, 1990.

[Han93] 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.

[HC86] D. A. Henderson and S. K. Card. Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Transactions on Graphics*, 5(3):211–243, July 1986.

[HC93] A. J. Hanson and R. A. Cross. Interactive visualization methods for four dimensions. In *IEEE Visualization '93 Proceedings*, pages 196–203. IEEE Computer Society, October 1993.

[HH91] J. Helman and L. Hesselink. Visualization of vector field topology in fluid flows. *IEEE Computer Graphics and Applications*, 11(3):36–46, 1991.

[HLL97] L. Hesselink, Y. Levy, and Y. Lavin. The topology of symmetric second-order 3d tensor fields. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):1–11, Jan-Mar 1997.

[ID90] A. Inselberg and B. Dimsdale. Parallel coordinates: a tool for visualizing multidimensional geometry. In *IEEE Visualization '90 Proceedings*, pages 361–378. IEEE Computer Society, October 1990.

[IFP96] V. Interrante, H. Fuchs, and St. Pizer. Illustrating transparent surfaces with curvature-directed strokes. In *IEEE Visualization '96 Proceedings*, pages 211–218. IEEE Computer Society, October 1996.

[IG97] V. Interrante and Ch. Grosch. Strategies for effectively visualizing 3d flow with volume lic. In *IEEE Visualization '97 Proceedings*, pages 421–424. IEEE Computer Society, October 1997.

[JL97] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. In *EUROGRAPHICS Workshop on Visualization in Scientific Computing Proceedings*, pages 57–66. Springer, April 1997.

[KB96] M.-H. Kiu and D. C. Banks. Multi-frequency noise for LIC. In *IEEE Visualization '96 Proceedings*, pages 121–126. IEEE, October 1996.

[Ker90] G. D. Kerlick. Moving iconic objects in scientific visualization. In *IEEE Visualization '90 Proceedings*, pages 124–129. IEEE Computer Society, October 1990.

[Laj94] T. Lajos. *Az áramlástan alapjai*. Müegyetemi Kiadó, Budapest, 1994.

[Lev91] H. Levkowitz. Color icons: Merging color and texture perception for integrated visualization of multiple parameters. In *IEEE Visualization '91 Proceedings*, pages 164–170. IEEE Computer Society, October 1991.

[LGWP96] H. Löffelmann, E. Gröller, R. Wegenkittl, and W. Purgathofer. Classifying the visualization of analytically specified dynamical systems. *Machine Graphics & Vision*, 5(4):533–550, 1996.

[LKG97a] H. Löffelmann, A. König, and E. Gröller. Fast visualization of 2D dynamical systems by the use of virtual ink droplets. In *Spring Conference on Computer Graphics 1997*, pages 111–118, June 1997.

[LKG97b] H. Löffelmann, Th. Kucera, and E. Gröller. Visualizing Poincaré maps together with the underlying flow. In *International Workshop on Visualization and Mathematics'97 Proceedings*. Springer, September 1997.

[LMG97] H. Löffelmann, L. Mroz, and E. Gröller. Hierarchical streamarrows for the visualization of dynamical systems. In *EUROGRAPHICS Workshop on Visualization in Scientific Computing Proceedings*, pages 203–209. Springer, April 1997.

[LMGP97] H. Löffelmann, L. Mroz, E. Gröller, and W. Purgathofer. Stream arrows: Enhancing the use of streamsurfaces for the visualization of dynamical systems. *The Visual Computer*, 13:359–369, 1997.

[LWW90] J. LeBlanc, M. O. Ward, and N. Wittels. Exploring n-dimensional databases. In *IEEE Visualization '90 Proceedings*, pages 230–239. IEEE Computer Society, October 1990.

[MCG94] N. Max, R. Crawfis, and Ch. Grant. Visualizing 3D velocity fields near contour surfaces. In *IEEE Visualization '94 Proceedings*, pages 248–255. IEEE Computer Society, October 1994.

[MSLG97] A. Milik, P. Szmolyan, H. Löffelmann, and E. Gröller. Geometry of mixed-mode oscillations in the 3-d autocatalator. Technical Report TR-186-2-97-14, Institute of Computer Graphics 186-2, Technical University of Vienna, Vienna, Austria, August 1997.

[MTS91] T. Mihalisin, J. Timlin, and J. Schwegler. Visualization and analysis of multi-variate data: A technique for all fields. In *IEEE Visualization '91 Proceedings*, pages 171–178. IEEE Computer Society, October 1991.

[PFTV88] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1988.

[PJS92] H.-O. Peitgen, H. Jürgens, and D. Saupe. *Chaos and Fractals*. Springer, 1992.

[PvW93] 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.

[PvWPS95] F. J. Post, Th. van Walsum, F. H. Post, and D. Silver. Iconic techniques for feature visualization. In *IEEE Visualization '95 Proceedings*, pages 288–295. IEEE Computer Society, October 1995.

[PW94] H.-G. Pagendarm and B. Walter. Feature detection from vector quantities in a numerically simulated hypersonic flow field in combination with experimental flow visualization. In *IEEE Visualization '94 Proceedings*, pages 117–123. IEEE Computer Society, October 1994.

[Rhe96] P. Rheingans. Opacity-modulating triangular textures for irregular surfaces. In *IEEE Visualization '96 Proceedings*, pages 219–225. IEEE Computer Society, October 1996.

[SH95] D. Stalling and H.-Ch. Hege. Fast and resolution independent line integral convolution. In Robert Cook, editor, *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 249–256, August 1995.

[SJM96] H.-W. Shen, Ch. R. Johnson, and K.-L. Ma. Visualizing vector fields using line integral convolution and dye advection. In *1996 Volume Visualization Symposium*, pages 63–70. IEEE, October 1996.

[SK97] H.-W. Shen and D. L. Kao. UFLIC: A line integral convolution algorithm for visualizing unsteady flows. In *IEEE Visualization '97 Proceedings*, pages 317–322. IEEE Computer Society, October 1997.

[SVL91] W. J. Schroeder, C. R. Volpe, and W. E. Lorensen. The stream polygon: A technique for 3d vector field visualization. In *IEEE Visualization '91 Proceedings*, pages 123–132. IEEE Computer Society, October 1991.

[TB96] G. Turk and D. Banks. Image-guided streamline placement. In *Proceedings SIGGRAPH '96*, pages 453–459, 1996.

[Tso92] A. A. Tsonis. *Chaos - From Theory to Applications*. Plenum Press, 1992.

[vW91] J. J. van Wijk. Spot noise: Texture synthesis for data visualization. In *Proceedings SIGGRAPH '91*, pages 309–318, July 1991.

[vW93a] J. J. van Wijk. Flow visualization with surface particles. *IEEE Computer Graphics & Applications*, 13(4):18–24, July 1993.

[vW93b] J. J. van Wijk. Implicit stream surfaces. In *IEEE Visualization '93 Proceedings*, pages 245–252. IEEE Computer Society, October 1993.

[WG97] R. Wegenkittl and E. Gröller. Fast oriented line integral convolution for vector field visualization via the internet. In *IEEE Visualization '97 Proceedings*, pages 309–316. IEEE Computer Society, October 1997.

[WGP97a] R. Wegenkittl, E. Gröller, and W. Purgathofer. Animating flow fields: Rendering of oriented line integral convolution. In *Computer Animation '97 Proceedings*, pages 15–21. IEEE Computer Society, June 1997.

[WGP97b] R. Wegenkittl, E. Gröller, and W. Purgathofer. Visualizing the dynamical behavior of wonderland. *IEEE Computer Graphics & Applications*, 17(6):71–79, December 1997.

[WLG97] R. Wegenkittl, H. Löffelmann, and E. Gröller. Visualizing the behavior of higher dimensional dynamical systems. In *IEEE Visualization '97 Proceedings*, pages 119–125. IEEE Computer Society, October 1997.