Vector Field Visualization on the Internet

U. Rauschenbach[*], H. Theisel,C. Perez Risquet[†]and H. Schumann
University of Rostock, Computer Science Department
PostBox 999, 18051 Rostock, Germany
theisel@informatik.uni-rostock.de

Abstract

We discuss the problems to be solved to develop a web-based vector field visualization system. Furthermore we present the system CurVis as one solution of these problems. As part of the CurVus system, two new global visualization techniques for vector fields are introduced: curvature plots and IDraw. Their combined application as hybrid technique is discussed as well. Finally applications and examples of the new techniques in the CurVis system are shown.

1 Introduction

With the availability of the World Wide Web, many applications which formerly required specialized software and powerful hardware can now be realized by placing that specialized software on a powerful server and allow access from cheap client computers using standard web browsers. Flow visualization techniques are a candidate for this approach, since they require powerful computers as well as specialized software.
The visualization of data is usually realized as a pipeline of processes, the *visualization pipeline*. If a visualization program is to be distributed between a client and a server, a decision has to be made where the visualization pipeline should be partitioned. Compared to standalone visualization systems, the client computers which access the visualization on the server have a variety of display capabilities, and data will have to be transmitted over networks of varying, often low bandwidth. To provide for each client the maximum possible quality of service, a WWW-based visualization service must be able to adapt to *context parameters* such as display size and bandwidth. Usually, the visualization session starts with an image showing the whole dataset. Later, the user often wants to explore interesting parts of the data set in greater detail. A *detail on demand* feature should allow to request greater detail in regions where it is needed. The user of a web-based visualization service should be able to submit his/her own data sets over the Internet.
Especially if the system is accessed using notebook computers with small screens, it is essential that the screen real estate is used efficiently by the visualization. Compared to local methods (e.g., streamlines or arrow plots), which provide information about the vector field only in some pixels of the resulting image, global methods (like LIC) are required

[*] Supported by the German Science Foundation under contract no. Schu-887/3-2.
[†] Supported by the German Academic Exchange Service.

which provide information about the underlying data in every pixel of the display. Thus it was part of the research presented in this paper to develop new global flow visualization techniques which are applicable - but not restricted - to visualization in the context of the Internet.
This paper is organized in the following way: chapter 2 discusses the problems to be solved to realize a web-based vector field visualization system and introduces the system CurVis (**C**FD **u**niversal **r**emote **Vis**ualizer) as a solution to the problems stated. Chapters 3 - 5 introduce new visualization techniques for vector fields which are appropriate for visualization in the internet context: curvature plots, IDraw and hybrid techniques.

2 CurVis - a System for Vector Field Visualization on the Internet

2.1 Pipeline Partitioning and Peephole Optimization

The visualization pipeline is a sequence of several processes converting data into an image. Brodlie[1] distinguishes between *filtering* of the data, *mapping* of the data to geometry and *rendering* of the geometry into a raster image. He classifies web-based visualization systems using the criterion between which processes the pipeline is partitioned.
Under the assumption that the visualization is accessed using web browsers, two solutions are possible. Most web browsers support some common raster image formats (GIF, JPEG) and the execution of Java applets. That's why the first solution is to compute an image on the server and to transmit it to the client, which corresponds to partitioning the pipeline between rendering and presentation. The second opportunity is to transmit the data set and a Java applet which computes the visualization locally, which corresponds to partitioning the pipeline between filtering and mapping. A third opportunity, partitioning between mapping and rendering, is not suitable for visualizing 2D flows. We investigated the first two opportunities using the *Integrate and Draw* method. As server, an SGI Power Challenge with 1GB main memory and six 196 MHz R 10000 processors has been used, the client was an 125 MHz Intel Pentium based PC with 32 MB main memory, connected to the server over a simulated slow Internet link with a net bandwidth of 9600 Bits/s. We broke the pipeline before the filtering step (using a Java applet and a binary which run locally on the client; the original data set has been compressed using gzip and fetched from the server) and between rendering and presentation (which is implemented by the CurVis system described below). Figure 1 shows the total times for these three partitioning alternatives for a wide range of data sets (bodden, water, cylinder, hyper and dipol). It is obvious that the partitioning decision between rendering and presentation is superior in all cases. For small data sets, the alternative to use a local binary is nearly as good, but offers less flexibility since this binary would have to be provided for each client platform.
That's why we decided to partition the pipeline between rendering and presentation.
In order to save transmission bandwidth, the peephole optimizer of the chosen partitioning must use image compression methods, since the transmitted data are image data. However, the images produced by the different visualization techniques require different com-

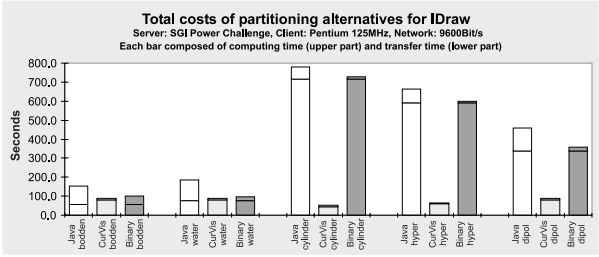

Figure 1: Partitioning alternatives for the Integrate and Draw method

pression methods. One could think that the lossy compression algorithm JPEG is always superior over lossless algorithms but that is not the case. As JPEG has been developed for continuous-tone true color images, it delivers best compression on this image class. On images with only a few colors, lossless compressors like GIF are often superior over JPEG. Since the different techniques generate images with very different characteristics, we are able to select the compression method best suited using our knowledge over these characteristics. Arrow plots and streamline images have only a few colors and large areas shaded in the background color, making them candidates for GIF. Curvature plots, IDraw and hybrid images, which show smooth gradients of color, can be better compressed using JPEG. Preferably, the progressive JPEG mode is used. Figure 2 provides a comparison. Using JPEG, a streamlines image can be compressed to the size of its GIF equivalent only by applying strong quantization which makes it impossible to interpret the visualization.

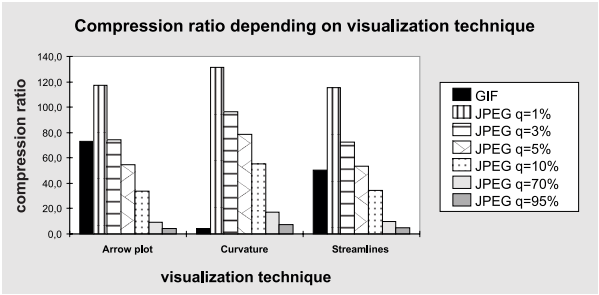

Figure 2: Suitability of GIF and JPEG for the compression of different flow visualizations

2.2 Context-based Image Generation

Context-based image generation can stretch the limits imposed by low bandwidth, restricted client display resources and processing power. It can save resources by generating images on the fly at a resolution and with a number of colors which can be displayed at client side. Compression of the generated image using the right compression method further decreases

the bandwidth demands. A method which supports progressive refinement of the image (e.g., progressive JPEG) or even only of selected regions of interest[6] allows the user to early assess the value of the visualization for his/her goals and to cancel unnecessary data transmissions. The adaptive image generation process is controlled by a set of *context parameters*:

- Display context: describes the client display size and possible number of colors
- Technique: selects the visualization technique
- Zooming: describes the mapping from the grid underlying the data to the grid underlying the pixel image to be generated
- Network: describes the available bandwidth

Depending on these context parameters, a visualization technique can be selected and parameterized by the system such that the image fits the client display capabilities.

2.3 System Overview

The system CurVis provides several visualization techniques: the classic *streamlines*, a simple *arrow plot*, the *curvature plot* technique presented in section 3, the *integrate and draw* method discussed in section 4 and *hybrid methods* proposed in section 5. Furthermore, the critical points of the vector field can be analyzed using the approach presented by Helman and Hesselink[3] and transmitted as a textual description. Each visualization module creates a bitmap image. More visualization modules can easily be added, as each module runs as a CGI program on the server. This allows the easy integration of arbitrary executables as long as they obey some simple interface specifications. When the image is generated, the context parameters discussed above are considered. An already generated image can be zoomed in by altering the zooming factor in the context parameter set. In this case, a new image is generated using the new context. Remote users can upload their data files for visualization to the CurVis server. CurVis automatically selects the image compression method best suited to the visualization generated. Figure 3 shows a block diagram of CurVis.

3 The Curvature Plot Visualization Technique

The curvature plot is a new global visualization technique for 2D flow fields. First introduced in [8], it gives a smooth image of the flow. The compression of these images gives usually gives good compression ratios; a reduced display a a notebook computer is in most cases sufficiant for displaying the visualization. This makes curvature plots an interesting candidate for the use in an Internet environment.
A 2D steady flow is usually described as a 2D vector field $V(x,y) = (u(x,y), v(x,y))^T$. A curve $\mathbf{s} \subseteq \mathbb{R}$ is called a *tangent curve* (stream line) of V if the following condition is satisfied: For all points $(x,y) \in \mathbf{s}$, the tangent vector of the curve in the point (x,y) has the same direction as the vector $V(x,y)$.

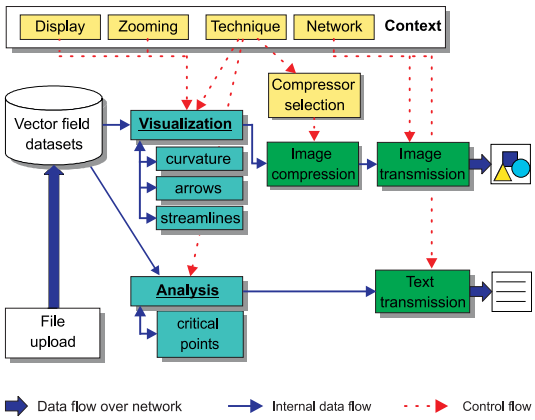

Figure 3: Block diagram of CurVis

For every point $(x, y) \in \mathbb{R}$ there is one and only one tangent curve through it (except for critical points of V, i.e. points with $\|V\| = 0$). Tangent curves do not intersect each other (except for critical points of V). They do not depend on the magnitudes but only on the directions of the vectors in the vector field.
Given a (non-critical) point (x_0, y_0) in V, let $\mathbf{s}$ be the tangent curve through (x_0, y_0). Furthermore, let $\mathbf{s}$ be parametrized in such a way that

$$\mathbf{s}(t_0) = (x_0, y_0) \quad (1)$$
$$\dot{\mathbf{s}}(t_0) = V(\mathbf{s}(x_0, y_0)). \quad (2)$$

($\dot{\mathbf{s}}(t)$ denotes the tangent vector of $\mathbf{s}(t)$). Then we can compute the second derivative vector $\ddot{\mathbf{s}}$ of $\mathbf{s}$ at t_0 by applying the chain rule to (2):

$$\ddot{\mathbf{s}}(t_0) = (u \cdot V_x + v \cdot V_y)(x_0, y_0). \quad (3)$$

Now we can easily compute the signed curvature of $\mathbf{s}$ in (x_0, y_0):

$$\kappa(t_0) = \frac{\det\left[\dot{\mathbf{s}}(t_0), \ddot{\mathbf{s}}(t_0)\right]}{\|\dot{\mathbf{s}}(t_0)\|^3}. \quad (4)$$

(2), (3) and (4) have the following consequence: in order to compute the curvature of a tangent curve in a certain point of a vector field it is not necessary to know the tangent curve itself. It is sufficient to know the vector field V and its partial derivatives.
Inserting (2) and (3) into (4), we obtain a simple formula for the curvature of tangent curves through every point of the vector field:

$$\kappa(V) = \frac{u \cdot \det[V, V_x] + v \cdot \det[V, V_y]}{\|V\|^3}. \quad (5)$$

(5) describes a scalar field in the domain of the vector field V. This scalar field describes the curvature of the tangent curve in every point of the domain. We call this scalar field

$\kappa(V)$ the *curvature of the vector field* V. $\kappa(V)$ is only defined for non-critical points. It does not depend on the magnitudes of the vectors in V.
The *perpendicular vector field* $V^\perp$ of a 2D vector field $V = (u, v)^T$ is defined as $V^\perp := (-v, u)^T$. For every point of the vector field, the vectors of V and $V^\perp$ are perpendicular to each other. We obtain for the curvature of $V^\perp$:

$$\kappa(V^\perp) = \frac{u \cdot \det[V, V_y] - v \cdot \det[V, V_x]}{\|V\|^3}. \tag{6}$$

We want to visualize the curvature κ of a 2D vector field in the following way: compute κ for every point of the domain and color code these values. To do this we use a continuous color coding map with the following properties: a negative value is mapped to a green color, a positive value is mapped to a red color. The higher the magnitude of the value the lighter the color gets. A zero value gives black; if the value diverges to plus (minus) infinity the red (green) color tends to white.

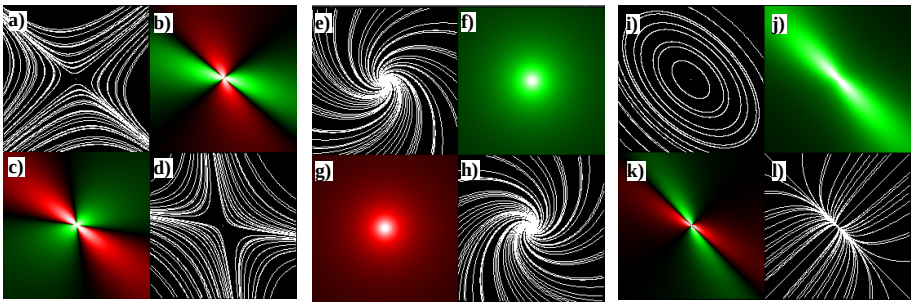

Figure 4: Linear vector field with saddle point (a..d); linear vector field with repelling focus (e..h); linear vector field with center (i..l)

The pictures a-d of figure 4 give an example of the vector field $V(x, y) = \begin{pmatrix} 1 \\ 3 \end{pmatrix} x + \begin{pmatrix} 4 \\ 1 \end{pmatrix} y$. This linear vector field has a critical point at $(0, 0)$ - a saddle point. Figure 4a shows a numerical tangent curve integration. Figure 4b is the visualization of its curvature. Figures 4d and 4c show the same for the perpendicular vector field $V^\perp$. In this case, $V^\perp$ has a saddle point at $(0, 0)$ as well. Note that generally the topology of a vector field and its perpendicular vector field might differ.
The reason for visualizing the curvature of both V and $V^\perp$ is shown by considering the following visualization properties:
In the curvature visualization b) of figure 4 the critical point appears as highlight. Considering (5), $\kappa(V)$ tends to infinity only if the denominator of κ tends to 0. This occurs only at critical points. Therefore, a highlight in the curvature visualization always indicates a critical point in the vector field. The reverse question arises: does every critical point produce a highlight in the curvature visualizations? The answer is yes, if we exclude certain degenerate points. A *degenerate critical point* of a vector field V is a critical point where

the directions of the vectors of V do not change in the neighborhood of the critical point. For non-degenerate critical points, we have the following

Theorem 1 *In the neighborhood of a non-degenerate critical point of a 2D vector field V, the curvature of V or $V^\perp$ (or both curvatures) tend to infinity.*

An exact definition of a degenerate critical point and the proof of this theorem can be found in [8]. The same theorem can be formulated in the following way: non-degenerate critical points in a vector field V always produce highlights in the visualization of the curvature of V or $V^\perp$.
Considering the curvature visualizations b) and c) of figure 4 again, another question arises: Do the curvature visualizations of V and $V^\perp$ contain all information of V ? The answer is given by

Theorem 2 *Given are two 2D vector fields V_1 and V_2 which have non-constant direction fields. If $\kappa(V_1) = \kappa(V_2)$ and $\kappa({V_1}^\perp) = \kappa({V_2}^\perp)$ then the directions of the vectors of V_1 and V_2 coincide in every point.*

See [8] for a proof. Theorem 2 has an interesting consequence: the curvatures of V and $V^\perp$ together contain all information about the directions of the vectors in V. Therefore, the curvatures of V and $V^\perp$ contain all information about the topology of V. This statement is true for vector fields of general topology.
Pictures e-h of figure 4 show a linear vector field with a repelling focus. Figure 4e is the numerical stream line integration, figure 4f is the curvature visualization. Figures 4h and 4g show the same for the perpendicular vector field. The repelling node appears completely green around the highlight in the curvature visualization and completely red in the curvature visualization of the perpendicular vector field. Figures 4 i-l show the visualization of a center. It appears completely green around the highlight in the curvature visualization (figure 4j) and has 4 different areas (colored red or green) each of 90 degrees in the perpendicular curvature visualization (figure 4k).
All critical points of figure 4 are of first order, i.e. they have $\det[V_x, V_y] \neq 0$ in the critical points. They can therefore be classified using the topology concepts described in [3]. Figure 4 also shows that the different kinds of critical points appear differently in the curvature visualizations. A detailed description of how to classify a first order critical point from the curvature visualization can be found in [9].
Figure 5 shows a collection of higher order critical points, i.e. points with $\det[V_x, V_y] = 0$. None of these points can be treated using the topology methods of [3] but their curvature visualization gives a fairly good impression of them. Figures 5 a-d show a saddle point with 4 pairs of tangent curves through it. In the curvature visualization (figure 5b) we have eight differently colored sections around the critical point. The perpendicular field (figure 5c) has eight different sections as well. Figure 5 e-h shows the visualization of the vector field $V(x, y) = (y^2, x^2)^T$ in the range $[-1, 1] \times [-1, 1]$. This vector field has a critical point with two elliptic sections in $(0, 0)$. Observing the stream line integration (figure 5e), this critical point may be missed. The curvature visualization (figure 5f) shows it clearly as a highlight with six differently colored sections around it. Here the visualization of the perpendicular curvature has two differently colored areas (figure 5g). Figures 5 i-l show the

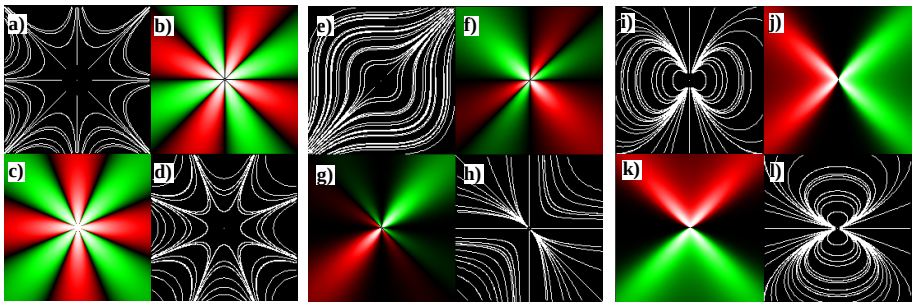

Figure 5: Higher order saddle point (a..d); critical point with two elliptic sectors (e..h); dipole (i..l)

visualization of a vector field describing a dipole. Both the visualization of its curvature (figure 5j) and its perpendicular curvature (figure 5k) show two differently colored sections around the highlighted critical point.
A general algorithm which infers the topology of higher order critical points from the curvature visualizations is still unknown. Nevertheless, the higher order critical points of figure 5 can be well distinguished from the first order critical points of figure 4 by their curvature visualizations.

4 The Integrate and Draw Visualization Technique

Another global visualization technique which was developed in the context of an Internet environment is Integrate and Draw (Idraw) ([5]). Idraw can be considered as an extension of the well-known Line Integral Convolution (LIC) method ([2], [7]).
Because of the nature of the LIC algorithm which averages pixel values along a field line, the resulting image tends to be muddy and difficult to see. A solution to this problem is to alter the length of convolution used in Cabral and Leedom's LIC algorithm ([2]). When longer convolution length are used, a higher number of pixels on a particular flow line are assigned to similar pixels color values. Unfortunately these results come at a performance cost "where doubling the length increases computation by a factor of four" ([4]).
The ideal output image would consist of a series of long thin distinct contrasting lines, depicting the streamlines. IDraw simply follows this idea and draws each field line with a different color. In the algorithm for each uncovered pixel in the output image, a new streamline is computed and a random color (a gray level between 0 and 255) is assigned to it. Then this streamline is mapped with its color onto the output image (see figure 6 left).
When two or more streamlines coincide at a pixel, their gray levels are simply averaged. The result of this idea is illustrated in Figure 6 (right). The resulting images are similar to LIC images, but faster to compute, because, as the name of the algorithm suggests, this method results in simply drawing streamlines without any convolution taking place.
Figure 6 (middle and right) illustrates the enhancements of IDraw images with respect to

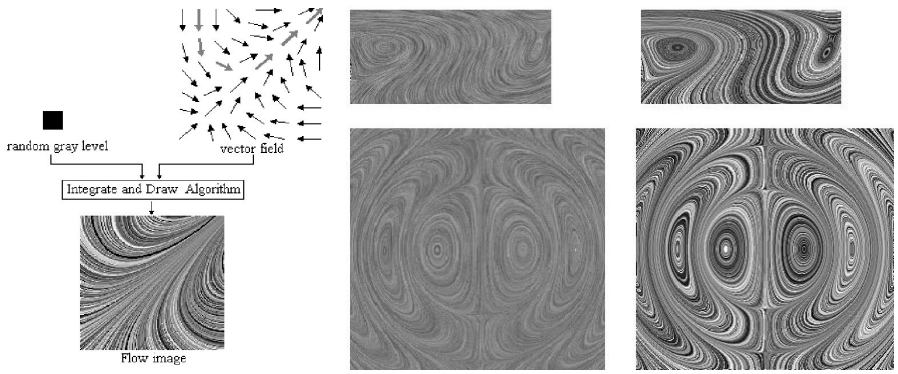

Figure 6: left: IDraw - the stream lines are mapped with random colors onto the output image; middle: LIC examples (flow around a cylinder,field if a dipole anntenna); right: IDraw examples

LIC images. By drawing more solid lines that do not undulate as frequently in value, the delineation of the flow lines has been improved. Thus, the image contrast has been increased and the vector field structure has became clearer. The Integrate and Draw algorithm can also be efficiently used for smooth detail enlargement. Using traditional LIC it is hard to visualize a vector field at different resolutions. It would require to use a resampled input texture as well as a resampled vector field. This can be easily accomplished with Integrate and Draw since it does not use any input texture. It is sufficient to use a smaller step size while integrating the stream lines. See [5] for more information about how smooth zooms can be accomplished with Integrate and Draw.

5 Hybrid Visualization Techniques

IDraw images are grey-scaled while the visualization of features like curvature give a color image. So it makes sense to combine both kinds of visualization techniques in order to more intuitive vector field visualization techniques. A simple linear interpolation between the IDraw image and the feature image gave satisfying results:

$$\mathbf{N} = (1 - t) \cdot \mathbf{I} + t \cdot \mathbf{F}$$

where $\mathbf{I}$ is the IDraw image, $\mathbf{F}$ is the feature image, and $\mathbf{N}$ is the newly created image. The parameter t can be used to emphasize either the one or the other image. Figure 7 illustrates the combination of IDraw and curvature plots.
Figure 8 shows two examples where other feature images were used. The left-hand image shows a combination of IDraw with a color coding of the flow direction in each point for $t = 0.5$. Shown is the flow in a bay area of the Baltic sea (Greifswalder Bodden). The right-hand image of figure 8 shows the combination of IDraw with a color coding of the vector magnitude in each point. Shown is the electrostatic field of a Benzene molecule.

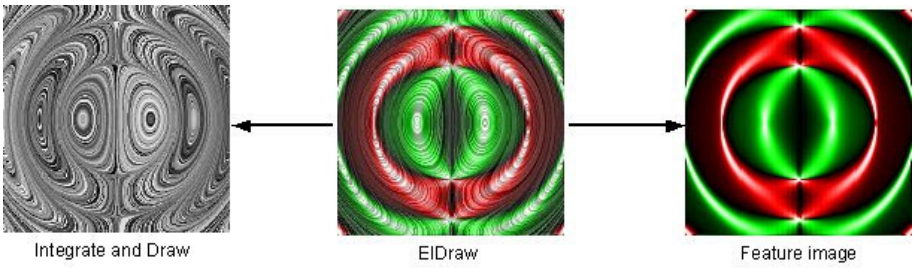

Figure 7: linear interpolation between Idraw (left) and curvature plot (right)

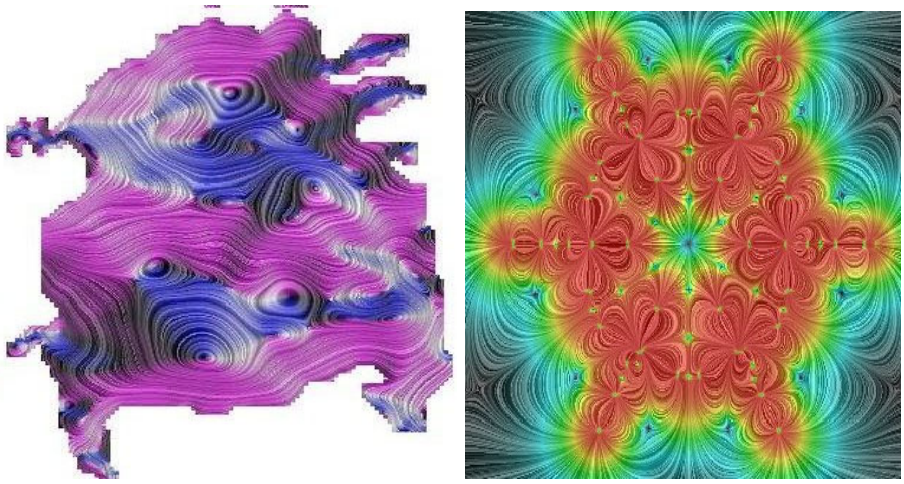

Figure 8: left: combination of IDraw and color coding of vector direction; right: combination of IDraw and color coding of vector magnitudes

6 Conclusions

We have presented the system CurVis for visualizing vector fields on the internet. As part of CurVis the new global visualization techniques curvature plots and IDraw were developed. Figure 9 shows two screendumps of the CurVis system.
CurVis can be accessed at
`http://www.informatik.uni-rostock.de/Projekte/movi/IIS/curvisrdr.html`

7 Acknowlegements

`The authors would like to thank the Konrad-Zuse-Zentrum fuer Informationstechnik, Berlin, Germany, and Dr. Kurt Frischmut (Rostock University) for providing the test data sets.`

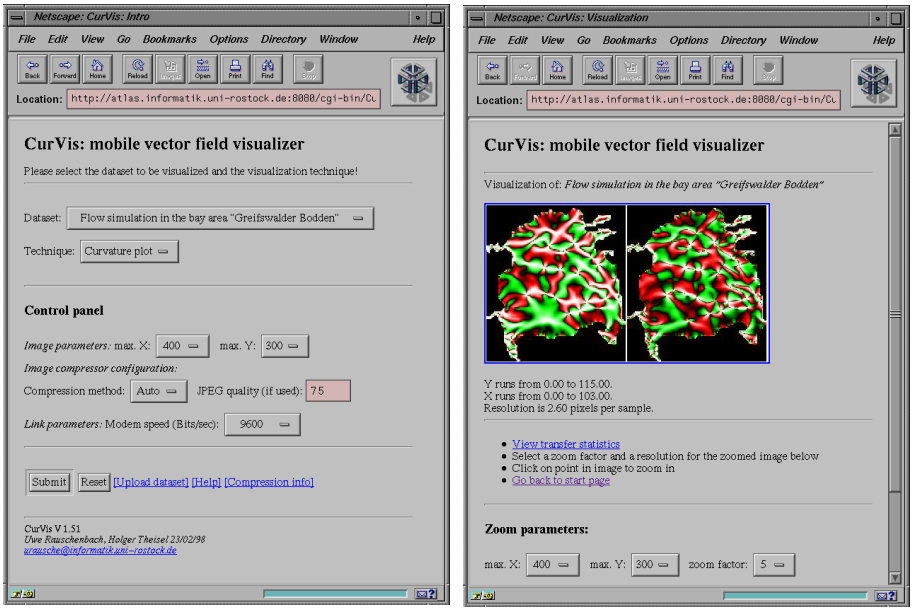

Figure 9: left: CurVis - start page; right: CurVis - curvature visualization

References

[1] Brodlie, K.: State of the Art Report: Cooperative Visualization, in: Proc. EUROGRAPHICS 98, Lisbon, Portugal, 1998.

[2] B. Cabral and L. Leedom. Imaging vector fields using Line Integral Convolution. *Proceedings Siggraph '93*, pp. 263-272, Computer Graphics 27, 1993.

[3] J.L. Helman and L. Hesselink. Representation and Display of Vector Field Topology in Fluid Data Sets. *IEEE Computer*, Vol. 22, No. 8, Aug. 1989, pp. 27-36.

[4] A. Okada and D.Lane. Line Integral Convolution with flow feature detection. *NAS Technical Reports*, NAS-96-007,1996.

[5] Perez Risquet, C.: Visualizing 2D Flows: Integrate and Draw, in: Bartz, D. (ed.): Proceedings of 9th EUROGRAPHICS Workshop on Visualization in Scientific Computing, Blaubeuren, April 19-22, 1998.

[6] Rauschenbach, U.: Progressive Image Transmission using Levels of Detail and Regions of Interest, in: Proc. IASTED Conf. on Computer Graphics and Imaging - CGIM 98, Halifax, Nova Scotia, Canada, June 1-4, 1998

[7] D. Stalling and H. Hege. Fast and Resolution Independent Line Integral Convolution. *Proceedings Siggraph '95*, Los Angeles, pp. 249-256, 1995.

[8] H. Theisel. *Vector Field Curvature and Applications*. PhD thesis, Dept. of Computer Science, University Rostock, Germany, 1996,
also available at http://wwwicg.informatik.uni-rostock.de/~theisel/

[9] H. Theisel. *Obtaining the Topology of Bilinear Vector Fields from their Curvature Visualizations*. Rostocker Informatik-Berichte 21, ISSN 0233-0784, 1998,
also available at http://wwwicg.informatik.uni-rostock.de/~theisel/