Strategies for Interactive Exploration of 3D Flow Using Evenly-Spaced Illuminated Streamlines

Oliver Mattausch*, Thomas Theußl*, Helwig Hauser†, and Eduard Gröller*

*Institute of Computer Graphics and Algorithms
Vienna University of Technology, Austria

†VRVis Research Center, Vienna, Austria

Abstract

This paper presents several strategies to interactively explore 3D flow. Based on a fast illuminated streamlines algorithm, standard graphics hardware is sufficient to gain interactive rendering rates. Our approach does not require the user to have any prior knowledge of flow features. After the streamlines are computed in a short preprocessing time, the user can interactively change appearance and density of the streamlines to further explore the flow. Most important flow features like velocity or pressure not only can be mapped to all available streamline appearance properties like streamline width, material, opacity, but also to streamline density. To improve spatial perception of the 3D flow we apply techniques based on animation, depth cueing, and halos along a streamline if it is crossed by another streamline in the foreground. Finally, we make intense use of focus+context methods like magic volumes, region of interest driven streamline placing, and spotlights to solve the occlusion problem.

Keywords: 3D flow visualization, illuminated streamlines, interactive exploration, focus+context visualization

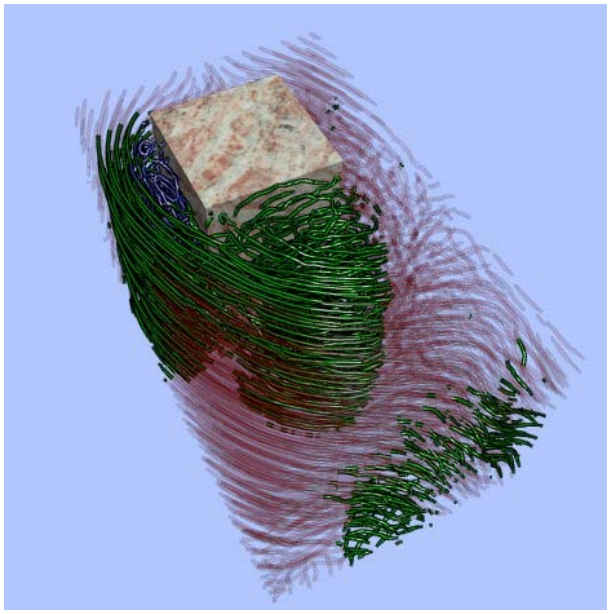


Figure 1: Flow around a block with high and low pressure coded as different colors and transparencies.

1 Introduction

Many convincing techniques for visualizing 2D flow are available. Visualization of 3D flow, on the other hand, is much more demanding. Several nontrivial issues have to be dealt with, like occlusion of important flow features, spatial perception, and orientation difficulties. Many methods need prior knowledge of the flow to achieve reasonable results, others show only details of the flow but not the general picture.

Thus, interactive exploration of the flow field is an important feature. We present a visualization tool that makes it possible to render an overview of the flow without having any prior knowledge of the important flow features at first, gaining fast insight of the flow, and then focusing on the interesting regions in a continuous manner. We use illuminated streamlines proposed by Zöckler et al. [19] as graphical objects representing the flow, because they provide fast rendering rates by exploiting current graphical hardware. Illuminated streamlines as a sparse representation give a good overview of the flow. Applying the Phong shading model for illumination to the streamlines improves the perception of spatial orientation and depth ordering.

The seeding strategy we chose is the evenly-spaced streamlines algorithm by Jobard and Lefer [8]. It gives

*{matt|theussl|meister}@cg.tuwien.ac.at, http://www.cg.tuwien.ac.at/home/
†Hauser@VRVis.at, http://www.VRVis.at/

Copyright © 2003 by the Association for Computing Machinery, Inc.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions Dept, ACM Inc., fax +1 (212) 869-0481 or e-mail permissions@acm.org.
© 2003 ACM 1-58113-861-X/03/0004 $5.00

an even distribution of streamlines and therefore produces a good overview of the flow without the need to specify a special seeding region. Although originally intended for 2D flow visualization, it can straightforwardly be extended to 3D. It was further extended to a multi-resolution model by Jobard and Lefer in 2001 [9]. For our visualization tool, this multi-resolution approach allows smooth transitions of focus+context regions and also a nearly continuous mapping of flow features to streamline density.

Our contribution to 3D flow visualization is a framework which provides:

- an automatic overview of the whole 3D flow without user input for exploration purposes with possibilities for subsequent analysis of the flow
- full interactivity after a few seconds of preprocessing
- flexible streamline appearance modifications for additional scalar data and better spatial perception
- interactive halos to greatly improve depth perception of the streamlines
- mapping of scalar values to streamline density as an automatic focusing strategy
- spotlights for user driven focusing

2 Related Work

In 2D, texture-based methods producing a dense representation of the flow, first introduced by van Wjik [17] with the spot noise algorithm, have proven to be quite appealing and comprehensive. Cabral and Leedom [1] developed LIC (Line Integral Convolution) which results in similar images. LIC uses a white noise input texture that is convolved along streamlines.

Sparse representations of vector fields, on the other hand, need a good seeding strategy for graphical primitives like streamlines or streak-lines to give visually pleasing results. Turk et. al [16] introduce an algorithm to produce hand drawn-style images of long evenly distributed streamlines. Jobard and Lefer [8] get similar results in significantly less computational time with the additional advantage that the algorithm is straightforwardly extended to 3D. This algorithm basically takes a separating distance as input parameter and calculates streamlines which are at least this separating distance apart.

Van Wijk [18] developed a framework to produce a variety of different flow visualizations in 2D, like moving particles, moving textures, or streamlines. The method is interactive and in particular suited for unsteady flow.

Although dense texture-based methods work well in 2D, a dense representation of a 3D field suffers from severe occlusion and perception problems. Interrante and Grosch [7] combined LIC using a sparse input texture with traditional volume rendering methods to visualize 3D flow fields. Rezk-Salama et al. [13] tackle the occlusion problem of volumetric LIC with interactive cutting planes. Sabadello [14] improves 2D and 3D spot noise with depth cueing and transparency coded velocity. Both Interrante and Sabadello make use of color coded scalar flow properties to further enhance spatial perception. A drawback of both methods is that they lack interactivity, an important feature of 3D flow visualization. The method of Rezk-Salama on the other hand is interactive but only shows the flow on planes and does not give a complete overview.

Crawfis and Max [2] investigated special vector filters to render a single scalar field and a vector field in the same image, showing the relationship of a scalar field and the flow in global climate models. The same authors extended the splatting algorithm by using texture map splats not only to render scalar volumes but vector volumes as well [3]. By exploiting texture hardware flow animation is done at interactive rates. Max et al. [10] introduce flow volumes that can be distorted in real time to render smoke.

Fuhrmann and Gröller [5] use animated dash tubes to show flow in a virtual environment. Tools useful for the exploration of the flow by focusing on a region of interest are presented (magic lens, magic box). We extend the magic box idea and achieve a similar level of interaction without the need of a virtual environment in our work. Guthe et al. [6] point out the use of animation to improve depth perception. They introduce an algorithm for animating texture based particles along a streamline. Flow features are mapped to particle velocity, form, and color.

Zöckler et al. [19] propose illuminated streamlines for renderings of sparse representations of a 3D vector field. Only line segments are used to draw the streamlines integrated from the underlying vector field. Implemented entirely in hardware, interactive frame rates for even a high number of streamlines are possible. This technique makes use of the OpenGL texture matrix function to simulate Phong shading. We adopted the illuminated streamlines as graphical objects for our work and made several enhancements to the basic form of the streamlines. By combining the illumination technique with projective texturing, spotlights can be used as light source, providing a very natural focusing method.

Our seeding strategy is an extension to the evenly-spaced streamlines algorithm by Jobard and Lefer [9]. It calculates multi-resolution streamlines for a number of separating distances beginning with the biggest separating distance (coarsest resolution) and ending with the smallest separating distance (most detailed resolution). This seeding strategy gives a good overview in a user defined level of detail on the one hand. On the other hand it allows us not only to effectively implement focus+context tools but also to map in real time scalar properties to streamline density.

Post et al. [11] provide a comprehensive overview of flow visualization state-of-the-art with an emphasis on flow feature extraction.

3 Basic 3D Algorithm

The basic evenly-spaced streamlines algorithm and the multi-resolution extension in 2D is described in detail by Jobard and Lefer [8, 9]. Because it is essential for our work, we shortly revise both algorithms here. Note that for both the basic and the multi-resolution model we only store the calculated streamlines as a list of sample points. This not only saves memory, but we also have maximum flexibility of controlling the appearance of the rendered streamlines, e.g., their color, texture, opacity, the number of actually displayed streamlines, or what portions of the streamlines are drawn.

3.1 Evenly-spaced streamlines

First we integrate a starting streamline. The starting streamline is placed into a streamline queue Q. We repeat the next steps until Q is empty:

- Take a streamline out of the queue.
- Collect all possible seed points in a distance d_{sep} from this streamline. In 2D there are only two possibilities per sample point, in 3D we choose them in such a way that they are d_{sep} apart from each other around the streamline, i.e., six of them.
- Integrate the streamlines from these seed points and put all valid streamlines into the queue (valid means long enough).

For the integration of longer streamlines, another parameter p_{test} is introduced. This parameter lies between 0.0 and 1.0. Streamlines are integrated until they come $p_{\text{test}} * d_{\text{sep}}$ close to another streamline. Smaller values of p_{test} produce longer streamlines.

3.2 Multi-resolution streamlines

Optionally, Q may be initialized with any number of streamlines as well. In the multi-resolution model, we first set d_{sep} to a rather big value, achieving streamlines for a very coarse representation of the vector field. Let us denote this resolution as res_0. Q is now initialized with the streamlines of res_0. For a slightly smaller d_{sep}, we apply the algorithm again, yielding the streamlines of res_1. This is repeated until we compute the most detailed resolution for the smallest d_{sep}. We now have streamlines for an arbitrary number n of different resolutions with $res_i \subseteq res_{i+1}$. A streamline of a resolution res_i is also a member of any resolution res_j with $i \leq j < n$. Note that streamlines of every res_i still suffice the evenly-spaced property with respect to the corresponding d_{sep}.

We observed that a high resolution number n has the drawback of shorter streamlines in high resolution levels, because the new streamlines are likely to be surrounded by lower resolution streamlines. Thus we made a slight modification to the introduced multi-resolution algorithm: To avoid that the streamlines become too short, we also reduce the second distance parameter p_{test} by a user-defined amount for every new resolution level.

3.3 Illuminated streamlines

To render illuminated streamlines, one could use cylinders to draw the streamline integral objects and light them with a graphics library like OpenGL. Alternatively, simple line segments as graphical primitives would reduce geometric complexity and therefore speed up rendering considerably. Unfortunately line segments have no distinct normal vector. Thus it is impossible to directly apply a shading model like Phong shading for the illumination in a streamline point p. The diffuse and specular term of the Phong equation require a normal vector:

$$I_p = c_{\text{amb}} + c_{\text{diff}} \vec{L} \cdot \vec{N} + c_{\text{spec}} \left(\vec{V} \cdot \vec{R} \right)^n$$

Choosing the normal vector $\vec{N}$ as the one that is coplanar to the tangent vector $\vec{T}$ in p and the light vector $\vec{L}$, $\vec{L} \cdot \vec{N}$ can be expressed without $\vec{N}$ [19]:

$$\vec{L} \cdot \vec{N} = \sqrt{1 - \left(\vec{L} \cdot \vec{T} \right)^2} \tag{1}$$

$\vec{V} \cdot \vec{R}$ can be rewritten without $\vec{R}$ in a similar way.

$$\vec{V} \cdot \vec{R} = \left(\vec{L} \cdot \vec{T} \right) \left(\vec{V} \cdot \vec{T} \right) - \sqrt{1 - \left(\vec{L} \cdot \vec{T} \right)^2} \sqrt{1 - \left(\vec{V} \cdot \vec{T} \right)^2} \tag{2}$$

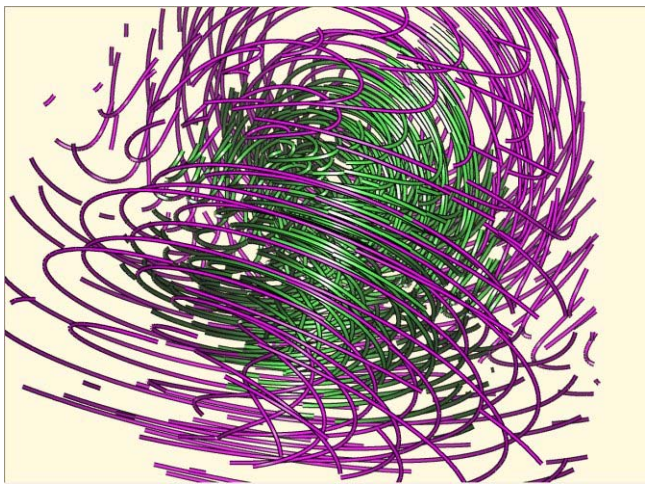
To exploit graphics hardware for the illumination, the texture matrix is loaded with $\vec{L}$ and $\vec{V}$:

$$\mathbf{X} = 1/2 \begin{pmatrix} L_1 & V_1 & 0 & 0 \\ L_2 & V_2 & 0 & 0 \\ L_3 & V_3 & 0 & 0 \\ 1 & 1 & 0 & 2 \end{pmatrix}$$

Now we set $\vec{L} \cdot \vec{T} = 2t_1 - 1$ in (1) and $\vec{V} \cdot \vec{T} = 2t_2 - 1$ in (2). With t_1 and t_2 running from 0.0 to 1.0 in both coordinates, the results of the equations are stored in a 2D texture map. Next we set the texture coordinates of p to the normalized tangent vector. Thus OpenGL calculates the inner products of the Phong equation with the help of the texture matrix, yielding the correct illumination color in p as texture color. See Zöckler et al. [19] for more details.

4 Streamline Enrichment

In this section we extend the original illuminated streamlines algorithm [19] to achieve more expressive results with thick streamlines, to map scalar flow features on the streamlines, to show direction of the flow, and to enhance the spatial perception of the streamlines.



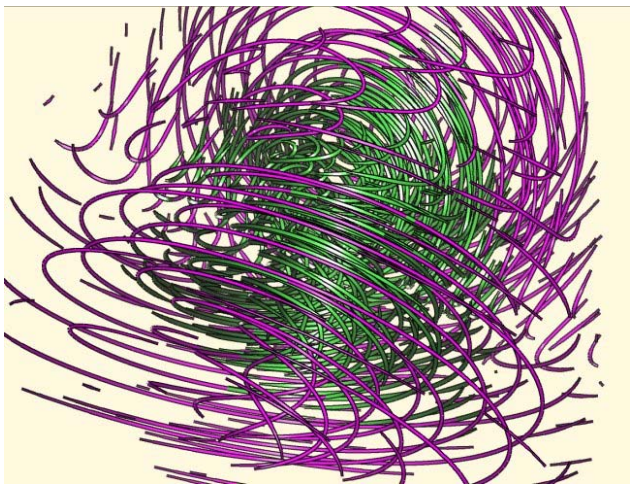


Figure 2: Lorenz system with focus region without end tapering (left) and with end tapering (right). Note the un-pleasing streamline endings at the focus region borders and streamline endings in general in the left image

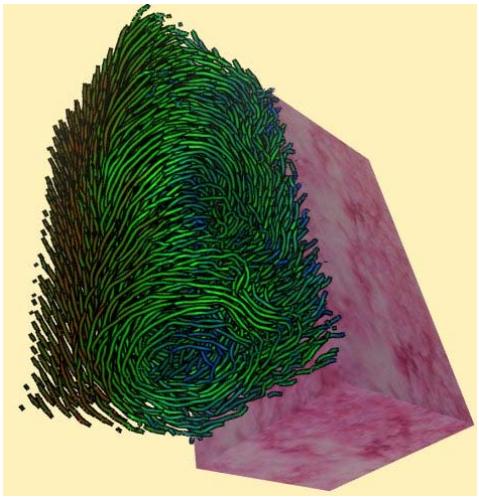


Figure 3: Color coded velocity with a continuous transfer function in the vicinity of the block.

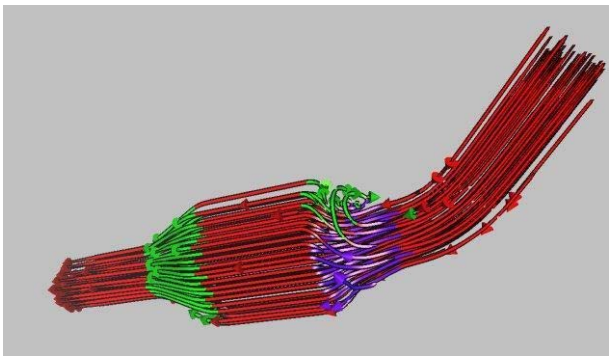


Figure 4: Catalytic converter with z-orientation as three scalar regions. Note the turbulence where the flow enters the converter.

4.1 Tapering

In 2D, tapering as proposed by Jobard and Lefer [8], i.e., decreasing streamline width whenever streamlines approach each other, results in hand-drawn style streamlines. In 3D, this approach is not applicable since streamlines will generally overlap in image space. However, if the width of the line segments is several pixels wide in image space, abrupt streamline endings do not look pleasing. By tapering the streamline endings, the streamlines look much more like real 3D objects. See figure 2 for the difference between tapered and un-tapered streamlines.

4.2 Scalar flow feature mapping

The scalar flow feature map works on a per sample point basis. For every streamline sample point, the scalar feature s is evaluated for the point. According to some function, the scalar value v in that point is mapped to streamline properties, e.g., line width, color, and opacity and even streamline density. s is one of many scalar flow features like velocity, vorticity, pressure, acceleration, or curvature.

In our implementation, we have two different ways to map scalar properties to streamline appearance and streamline density. The first is a piecewise linear function, defined by three property points, where the first one represents the streamline properties at the minimum value of scalar s, the second one the properties for a value v between the minimum and maximum value, the third represents the maximum scalar value property. Then the properties are linearly interpolated between the three property points. This type of mapping function is suitable for smooth transitions between regions with highly different values of a scalar s.

For the second function, the value range of scalar s can be split up into different streamline property regions,

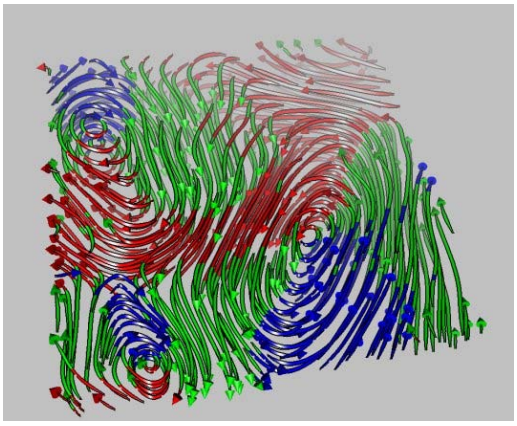


Figure 5: Smog over Europe, orientation in y-axis coded with 3 scalar interval region

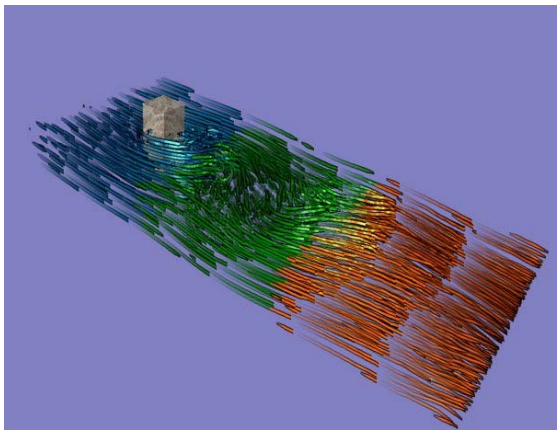


Figure 6: Flow around block with opacity function showing direction. Position in y direction is coded as scalar regions.

where every region represents an interval of scalar values. Streamline sample points where the value of s falls into a the scalar range interval of r receive all streamline properties of r. This function should be used in cases where we want to emphasize small differences between scalar values, or when we wish a clear separation of regions with different scalar values. Figure 3 shows velocity mapped with a continuous and figure 1 pressure with a scalar interval region function. Figure 4 shows a rendering with z-orientation mapped to scalar flow feature intervals. Figure 5 shows a rendering of the smog dataset where streamline orientation is mapped to three scalar interval regions.

4.3 Flow direction mapping

To show direction of the 3D flow we add a cone on the end of each streamline. Because we use only one cone per streamline in this case, one of the main advantages of streamlines in real-time rendering, namely the fast rendering time due to the not very complicated geometry, is still intact with the benefit of getting a clue of flow direction. Alternatively, arrow-like cones can be mapped on the streamline in periodic distances, say every n streamline points. This still does not affect performance much if the number of cones is low, but it has the additional advantage of giving a hint of velocity if point distances are directly proportional to the velocity of the flow in this region.

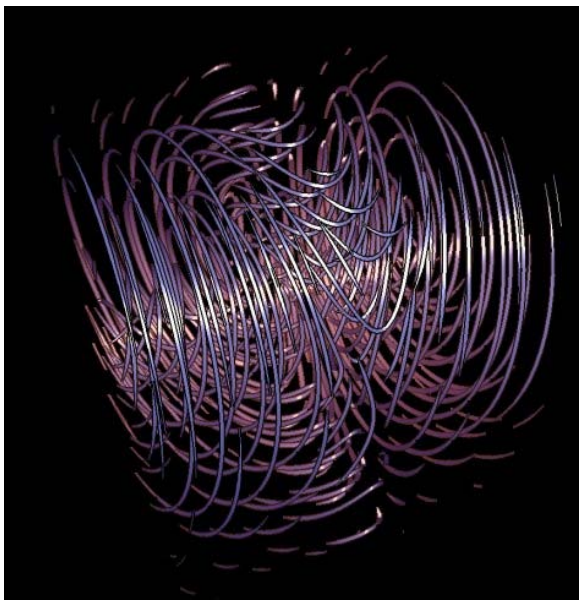


Figure 7: The Lorenz system with color coded depth

Alternatively, Zöckler et al. [19] present an opacity function that produces particle like objects which also shows flow direction (see figure 6).

4.4 Enhancing the spatial perception

Spatial perception is a major issue in 3D flow visualization. The image often looks quite flat and lacks a 3D impression if lines are used for rendering, and it is hard to tell what is in the foreground or in the background. Interrante et al. [7] already investigated this problem. They propose depth cueing, animation, and streamline color as methods to clarify the overall depth relations and present visibility impeding halo regions that enclose each streamline. That halo regions create gaps on the streamline crossing behind another streamline in the foreground. This is a natural means to show depth continuities which is often used by artists. In this section, we discuss the integration of these methods of enhancing depth perception in our framework.

4.4.1 Depth cueing and color-coded depth

Depth cueing is a simple, effective way to enhance the depth impression. With OpenGL, depth cueing can easily be done with glFog. The typical depth cueing effect is achieved if the color of glFog is the same as the background color. A color different to the background color gives an alternative depth effect like in figure 7. Another approach of using color to show depth is the additional

scalar mapping functionality to map position to streamline color or material. Unlike depth cueing, the colors do not depend on image z-distances and thus stay constant for each streamline during rotation. This method gives a good clue for understanding the spatial relationships in the flow. Figure 6 shows color coded direction.

4.4.2 Interactive view-point changes

Animation helps a lot to get a 3D impression. Simply moving or rotating the vector field already serves this purpose, This can be observed when the animation is stopped, since the image immediately appears flat. Since all the techniques presented in this paper are interactive, animation is possible at any time and greatly improves depth perception.

4.4.3 Interactive halos

Halos are easily implemented for illuminated streamlines using OpenGL. The depth buffer must be enabled with glDepthFunc set to GL_LESS. We first render a streamline, then we draw a black streamline with greater line width exactly on the same vertex positions. Because fragments with equal depth that come second are rejected, the pixels of the black streamline are drawn only left and right of the original streamline, acting as a black border on each side that is always parallel to the screen from every viewing angle. When this border region crosses another streamline in the foreground, the halos appear on the background streamline. Besides from enhancing depth impression, the halos give a more plastic 3D look to the streamlines. This is similar to silhouette drawing in non photo-realistic rendering [12]. If streamlines are drawn semi-transparent, the halo is drawn with the same transparency value. See figure 8 for the difference between streamlines with and without halos.

5 Focusing Strategies

Occlusion is a big problem of 3D flow images. Streamline density and/or opacity have to be adjusted carefully to extract the important information from the flow. Therefore it is useful to have an overview of the flow with a low streamline density and maybe low opacity. Interactive tools are required to specify geometric objects which define regions, usually called magic volumes, where interesting parts of the flow are drawn in great detail. Fuhrmann and Gröller [5] use magic lenses and magic boxes for exactly this purpose.

5.1 Magic volumes

We extended the magic box approach of Fuhrmann and Gröller [5]. First we implemented and tested different focus regions: cubes, rectangular prisms, and spheres. While cubes and spheres have only one degree of freedom, the rectangular prism has three, allowing the user to adjust the shape of the focus much better. Figure 9 shows a rectangular prism as magic volume region.

When starting in focus+context mode, the streamline density in the context can be reduced continuously to any resolution smaller or equal the focus resolution. Because the streamlines are already calculated, all transformations affect only the amount of streamlines drawn on screen and can therefore be done in real time. With the multi-resolution streamlines, streamlines of lower resolutions are always member of all higher resolutions. Thus context streamlines are never clipped at the focus region while preserving the proper density of evenly distributed streamlines. Along with a proper tapering of streamline endings as presented in section 4.1, changing the size of the focus region results in a smooth transition which looks like natural streamlines growing and shrinking. Not only a smaller streamline density, but also more transparency and/or smaller line width in the context helps to solve the occlusion problem. In our implementation, it is possible to adjust these parameters in real time to any percentage of the corresponding focus parameters.

To put further emphasis on the focus region, the focus and context can be assigned a different texture material. Additional information is only displayed in the focus.

5.1.1 Magic volume boundary region

Another issue is the focus boundary region. A sharp boundary region where the parameters jump from one region to another discontinuously often is not very visually appealing, especially if the opacity or line width difference between focus and context is high. This effect is smoothed out by specifying a boundary region with a user-definable transition width. If a streamline point is not directly in the focus but falls in this transition region, its distance to the focus is then taken to linearly interpolate line width, opacity, and streamline density between focus and context values [4]. Figure 10 shows a binary and a continuous focus boundary region.

5.1.2 Magic volume as a seeding region

The magic volume also has a seeding region and magnification functionality. If the user has investigated all the available details in the focus region, but wants to gain even more insight into the flow, the focus boundaries can be made the new field boundaries. The streamlines are then recalculated with an even smaller d_{sep}. In this new field, a new magic volume can be used to further magnify the flow.

In a second mode, magic volumes act as a seeding region where new streamlines are generated (see figure 11). The streamlines are not allowed to be seeded outside of the magic volume, but are allowed to grow outside of the magic volume. For the very flexible magic rectangular prism, some interesting seeding regions can be thought of,

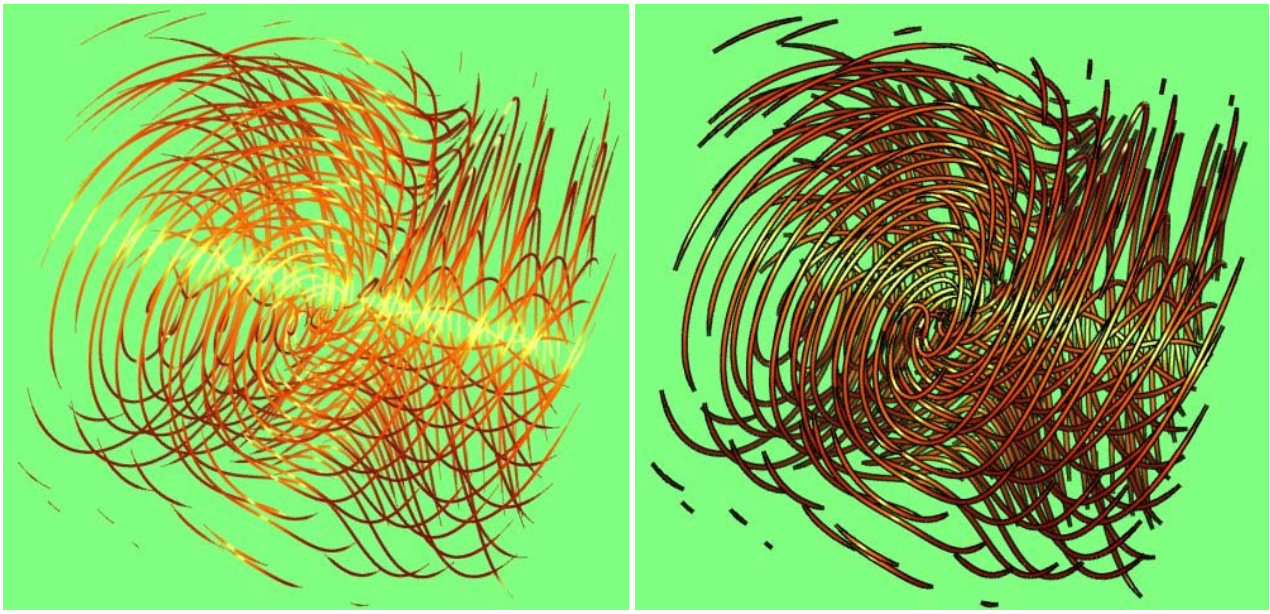


Figure 8: The Lorenz system without halos (left) and with halos (right).

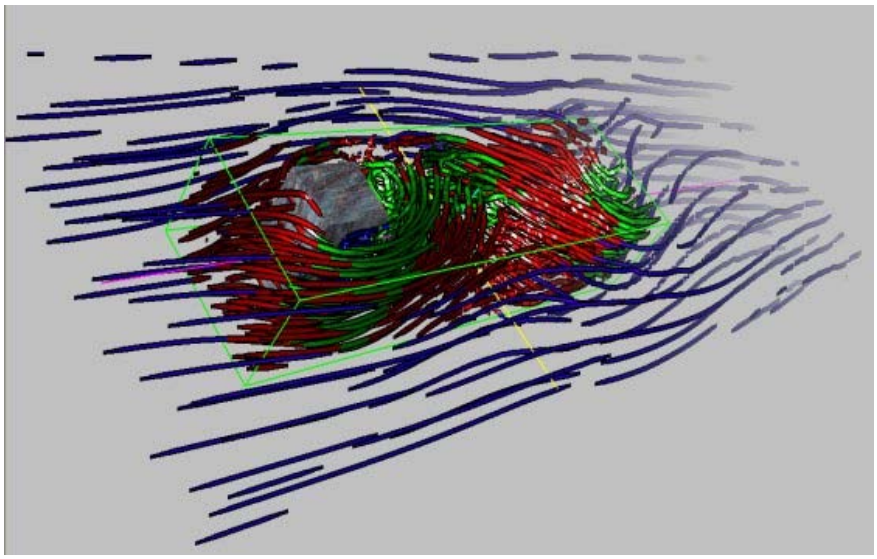


Figure 9: Flow around block with rectangular prism as magic volume (volume shown as geometry) and color coded pressure

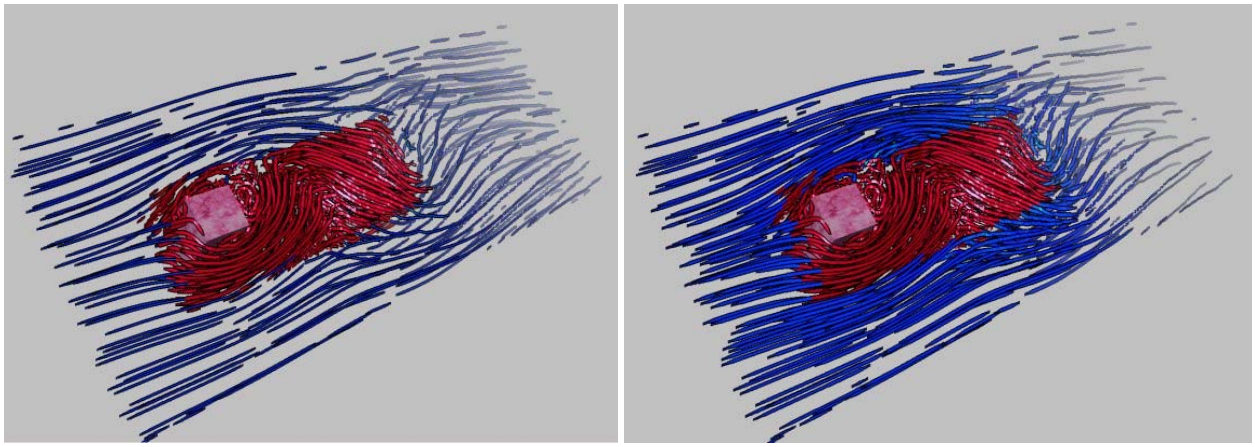


Figure 10: Flow around block with sharp border magic volume (left) and smooth border magic volume (right).

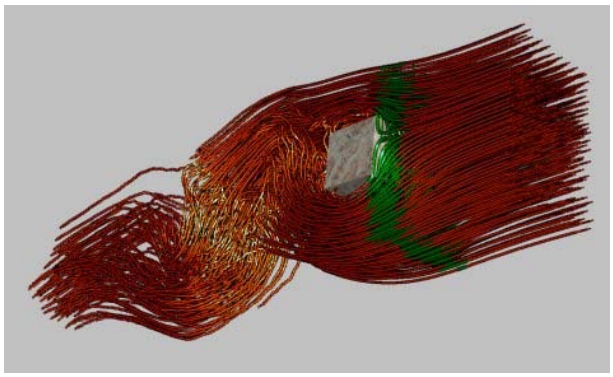


Figure 11: Magic volume used as seeding region (drawn green) in streamline calculation for the flow around block

like flat patches or line-like long and thin objects. This seeding method together with a quite small d_{sep} visualizes where streamlines originating from the starting region go.

5.2 Mapping of scalar features to streamline density

Our scalar property mapping method is very flexible as it can be done for any scalar feature type s and can map the scalar values v quite freely to any streamline property. It may be sufficient in 2D to map the scalar values to streamline width or streamline color. In 3D we have an occlusion problem when the streamline density is too high, while we might miss some details if streamline density is too low. Favorably, regions of high interest, for example regions with high velocity, are represented in great detail, whereas not so interesting regions are represented with low detail, so they do not occlude the more interesting parts of the flow. Of course, we can map scalar values to a function of streamline opacity. Then regions of high interest are drawn highly opaque and everything else more transparently (see figure 1).

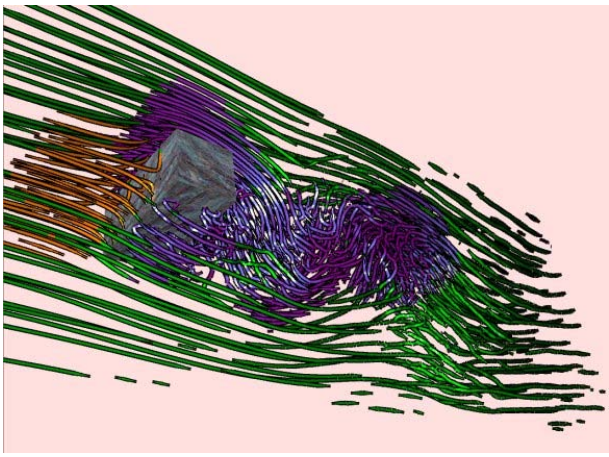


Figure 12: Flow around a block with three interval regions and pressure mapped to streamline density

A more intuitive region of interest representation than opacity is streamline density. Regions of highest interest should be represented by streamlines with the smallest d_{sep}, regions of lower interest with a distance $d_{\text{sep}} + \Delta d$. Δd is a positive factor that is indirect proportional to the "interest" of streamline sample points with a value v of scalar parameter s. v may be any scalar property like velocity, position, streamline orientation. v usually ranges from the lowest to the highest value of s among the sample points of all calculated streamlines. The only exception is streamline orientation in x,y or z direction, where the lower and upper border are always 0 and 2π, respectively. If, for example, regions of high velocity are interesting to us, Δd is 0.0 for the sample points of the flow with the highest velocity and grows for low velocity sample points. In a flow where regions of slow flow are more interesting, we let Δd grow directly proportional to the velocity. We do streamline integration with a fairly high streamline density in a preprocessing step, and the streamlines are then

Figure 13: Spotlight shining on the t-junction data set.

stored as sample points. As scalar functions are evaluated per sample point, changing the streamline density in some region is just a matter of displaying or not displaying sample points and thus an interactive operation.

In the multi-resolution model explained in section 3 streamlines of lower resolution are automatically included in all the higher resolutions, while streamlines of any resolution res_i are evenly-spaced with respect to the separating distance d_{sep_i}. Thus streamlines that grow from a region of lower resolution into a region of higher resolution are never eliminated, because they are also a member of the higher resolutions. To get the proper streamline density in all parts of the flow, we just have to render all streamline sample points that belong to the most detailed resolution having a separating distance $d_{\text{sep}_i} \geq d_{\text{sep}_n} + \Delta d$. If sufficiently many streamline resolutions are calculated, almost continuous transitions of streamline density can be done.

5.3 Spotlights

With spotlights we can light a region of the flow, while the rest remains dark or is lit with a dimmer light. It serves a similar purpose as the magic lens introduced before. Figure 13 shows the t-junction data set with the spotlight emphasizing the interesting region. However, spotlights are superior to magic lenses as introduced by Fuhrmann and Gröller [5]. They are really involved in 3D space, instead of the 2D based magic lens where the only 3D connection is the cutting plane of the lens. While the position of the magic lens has to be adjusted when the viewpoint changes, the spotlight still lights the same region. Spotlights also enhance the spatial perception of the image.

We implemented the spotlight with simple projective texturing. See Segal et al. [15] for more details. Alternatively, instead of a perspective viewing frustum an orthogonal viewing frustum may be used to project the spotlight texture on the streamlines. Then the light rays are parallel and the real world equivalent would be a flashlight.

6 Results

Please refer to `http://www.cg.tuwien.ac.at/research/vis/Miscellaneous/ESIS/` for high-resolution color images. In table 1 we present some statistics to back up our claim of interactivity. These results were produced on an Atlon 1.4GH computer with 512 MB Ram and a GeForce 3 graphics card. We achieved 6-30 frames per second (FPS) depending on the number of actually displayed sample points and even over 100 FPS for a low streamline resolution. Streamlines were integrated with the second order Runge-Kutta method. The number of resolutions had little effect on the preprocessing time since approximately the same number of streamlines and sample points is calculated. Halos reduced the number of frames per second slightly since only additional line segments are drawn. Drawing of arrows for depicting flow orientation had the largest effect on rendering time due to the real geometry of the cones (the measured times in table 1 correspond to one cone per streamline). Spotlights did not affect the rendering time at all.

7 Conclusions

We presented a visualization tool to explore and analyze 3D flow in real-time. The main advantage of this tool lies in its interactivity. Without any prior knowledge of the flow structure, the user can learn about important flow features within short time, then concentrate on the interesting regions in detail. An important feature are the focus+context methods. Magic volumes provide a powerful tool to investigate parts of the flow without the typical occlusion problems of 3D flows, while keeping up connection to the rest of the flow and not losing the overview. Spotlights have a long history as a tool used for focusing in real world. Our spotlight functionality not only serves as a focus tool, but also enhances the 3D impression of the image. For region of interest driven focus+context specification we introduced the mapping of scalar flow features to streamline density. Another contribution of our work is the enrichment of the basic illuminated streamlines algorithm. To make the streamlines visually more pleasing we use tapering and introduce interactive halos to enhance contrast and spatial perception, further we show flow direction with opacity functions and arrows.

8 Acknowledgments

We thank Wim de Leeuw for the flow around a block and smog datasets and Markus Hadwiger for the catalytic converter and t-junction datasets. This work was done as part of the FWF project AngioVis P15217 and the VRVis Research Center, which is funded by an Austrian research program called Kplus.

dsep	streamlines	sample pts	displ. pts	res.	preproc.	FPS	halos	arrows	spotlights	all
3% – 25%	1758	143602	76128	10	101 sec	11.8	9.5	6.9	11.8	6.3
3%	1567	160443	83112	1	135 sec	10.5	8.7	6.7	10.5	5.8
9%	141	18048	9061	1	2 sec	$\geq$100	$\geq$100	$\geq$100	$\geq$100	$\geq$100
3% – 25%	1089	115323	78831	10	57 sec	11.8	9.5	8.4	11.8	7.2
3%	922	119847	81709	1	51 sec	11.8	9.5	8.7	11.8	7.4
4.5% – 25%	481	51134	34482	10	15 sec	29.2	22.5	18.4	29.2	16.7
4.5%	373	51301	37657	1	12 sec	29.2	22.5	20	29.2	16.7

Table 1: The first block shows results for the Lorenz system sampled on a $100 \times 100 \times 100$ regular grid, the second block for the flow around the block data set $(280 \times 518 \times 30)$. The distance between streamlines (dsep) is given in percent of the largest dimension of the data set (100% meaning only one streamline). FPS means performance without halos, arrows, and spotlights, which are correspondingly switched on for the next three columns. The last column depicts performance for rendering with halos, arrows, and spotlights.

References

[1] B. Cabral and L. Leedom. Imaging vector fields using line integral convolution. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, pages 263–269, 1993.

[2] R. Crawfis and N. Max. Direct volume visualization of three-dimensional vector fields. *1992 Workshop on Volume Visualization*, pages 55–60, 1992.

[3] R. Crawfis and N. Max. Texture splats for 3D scalar and vector field visualization. In *Proceedings of IEEE Visualization '93*, pages 261–267, 1993.

[4] H. Doleisch and H. Hauser. Smooth brushing for focus+context visualization of simulation data in 3D. *Journal of WSCG*, 10(1):147–154, 2002.

[5] A. L. Fuhrmann and M. E. Gröller. Real-time techniques for 3D flow visualization. In *Proceedings of IEEE Visualization '98*, pages 305–312, 1998.

[6] S. Guthe, S. Gumhold, and W. Strasser. Interactive visualization of volumetric vector fields using texture based particles. *Journal of WSCG*, 10(3):33–41, 2002.

[7] V. Interrante and C. Grosch. Strategies for effectively visualizing 3D flow with volume LIC. In *Proceedings of Visualization '97*, pages 421–424, 1997.

[8] B. Jobard and W. Lefer. Creating evenly-spaced streamlines of arbitrary density. In *Visualization in Scientific Computing '97. Proceedings of the 8. Eurographics Workshop*, pages 43–56, 1997.

[9] B. Jobard and W. Lefer. Multiresolution flow visualization. In *WSCG 2001 Conference Proceedings (Posters)*, pages 33–37, 2001.

[10] N. Max, B. Becker, and R. Crawfis. Flow volumes for interactive vector field visualization. In *Proceedings of IEEE Visualization '93*, pages 19–24, 1993.

[11] F. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. Feature extraction and visualization of flow fields. In *State-of-the-Art Proceedings of EUROGRAPHICS 2002 (EG 2002)*, pages 69–100, 2002.

[12] R. Raskar. Hardware support for non-photorealistic rendering. In *SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pages 41–47, 2001.

[13] C. Rezk-Salama, P. Hastreiter, C. Teitzel, and T. Ertl. Interactive exploration of volume line integral convolution based on 3D-texture mapping. In *Proceedings of IEEE Visualization '99*, pages 233–240, 1999.

[14] M. Sabadello. Enhancing spot noise visualizations of 2d and 3d vector fields. In *Central European Seminar on Computer Graphics Proceedings*, pages 19–29, 2002.

[15] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, and P. Haeberli. Fast shadows and lighting effects using texture mapping. *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26(2):249–252, July 1992.

[16] G. Turk and D. Banks. Image-guided streamline placement. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 453–460, 1996.

[17] J. J. van Wijk. Spot noise-texture synthesis for data visualization. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, pages 309–318, July 1991.

[18] J. J. van Wijk. Image based flow visualization. In *Computer Graphics (SIGGRAPH 2002 Proceedings)*, pages 745–754, 2002.

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