Interactive Exploration of Volume Line Integral Convolution Based on 3D–Texture Mapping

C. Rezk–Salama, P. Hastreiter, C. Teitzel, T. Ertl

Computer Graphics Group
University of Erlangen–Nuremberg, Germany [*]

Abstract

Line integral convolution (LIC) is an effective technique for visualizing vector fields. The application of LIC to 3D flow fields has yet been limited by difficulties to efficiently display and animate the resulting 3D–images. Texture–based volume rendering allows interactive visualization and manipulation of 3D–LIC textures. In order to ensure the comprehensive and convenient exploration of flow fields, we suggest interactive functionality including transfer functions and different clipping mechanisms. Thereby, we efficiently substitute the calculation of LIC based on sparse noise textures and show the convenient visual access of interior structures. Further on, we introduce two approaches for animating static 3D–flow fields without the computational expense and the immense memory requirements for pre–computed 3D–textures and without loss of interactivity. This is achieved by using a single 3D–LIC texture and a set of time surfaces as clipping geometries. In our first approach we use the clipping geometry to pre–compute a special 3D–LIC texture that can be animated by time–dependent color tables. Our second approach uses time volumes to actually clip the 3D–LIC volume interactively during rasterization. Additionally, several examples demonstrate the value of our strategy in practice.

Keywords: Flow Visualization, Animated LIC, Direct Volume Rendering, 3D–Textures Mapping, Interactive Volume Exploration

1 Introduction

The visualization of 3D–flow phenomena is an important topic of research. Over the last few years this has led to a number of different techniques aiming at the meaningful analysis of vector fields. Traditionally, simple arrow plots are used, which directly show every vector with a respective graphical representation. More advanced methods use icons [24] allowing to integrate several parameters describing the field. However, these icons are sometimes difficult to interpret due to the unfamiliar way of representation and the complexity of the information. More sophisticated approaches depict the properties of a vector field by using methods like stream lines, stream surfaces [14], volume flows [22] and various techniques of particle tracing. Although the polygonal primitives used for these methods allow a fast manipulation of the 3D–representation, they are restricted to a rather coarse spatial resolution.

In order to overcome these limitations, texture–based approaches gained increasing attention since they take into account all the information of a data set. The introduction of line integral convolution *(LIC)* [4] significantly improved the visualization of 2D vector data. It is an effective and versatile technique for representing flow fields with small scale structures. Although a LIC volume is computed in the same way as a 2D–LIC image, the 3D–approach is rarely used. This is mainly related to difficulties in approaching interior structures of the data, which is fundamental for the analysis and interpretation of the vector field.

A completely different approach is employing volume visualization. In this context texture splats were introduced [6] in order to accelerate the visualization process by using hardware assisted 2D–texture mapping. Based on the original technique an extension was developed to process vector fields. Additionally, different volume rendering approaches are suggested in [9] to visualize the scalar components of computational fluid dynamics *(CFD)* data. Thereby, the analysis is supported by different functionality ranging from a simple illumination model to transfer functions applied to extract specific features. Due to the required load of calculations the approaches based on directly visualizing the data prohibit to manipulate the 3D–representation interactively.

As a solution to this bottleneck, modern high–end graphics workstations provide a large number of trilinear interpolation operations per second. Thereby, direct volume rendering is performed at high image quality and interactive frame rates [3] which greatly improves the spatial perception within a 3D–representation. However, the stream lines inside a 3D–LIC texture are too dense and intricate to visualize them as a whole. As proposed in [15] the application of sparse input textures enhances the visualization results.

In addition to the interactive manipulation guaranteed by 3D–texture mapping, we suggest a dedicated selection of supporting functionality. This is an essential prerequisite for the comprehensive exploration of flow fields using 3D–LIC representations. It includes the interactive and intuitive abilities to adjust transfer functions and to apply different clipping mechanisms.

Another effective way to enhance the visualization of static vector fields is the animation of LIC volumes. As a drawback, techniques which are used for 2D–LIC are less applicable in the 3D case. Above all, this includes the computation of a separate LIC texture for each time step, which results in a great computational expense and an immense amount of data. To avoid the performance penalty and the high memory requirements that come with loading and storing large pre–computed 3D–textures, we suggest not to animate the 3D–LIC itself, but to use an animated clipping object instead. In this context we introduce two different approaches, which both use a single 3D–LIC texture and a set of clipping objects. To display animated 3D flow at interactive frame rates direct volume rendering based on 3D–texture mapping is performed.

After a short survey about the basic ideas of LIC in section 2, direct volume rendering using 3D–texture mapping is briefly described in section 3. Subsequently, section 4 discusses appropriate settings of transfer functions which are interactively adjusted. In the same context we show how to produce meaningful semi–transparent representations which efficiently substitute the calculation of LIC based on sparse noise textures. Thereafter, section 5 presents different clipping mechanisms and explains how to use them effectively for the analysis of flow fields. Then, animating 3D–LIC using time–dependent color lookup tables (section 6) and

[*]Lehrstuhl für Graphische Datenverarbeitung (IMMD9),
Universität Erlangen–Nürnberg,
Am Weichselgarten 9, 91058 Erlangen, Germany,
Email: `rezk@immd9.informatik.uni-erlangen.de`,
URL: `http://www9.informatik.uni-erlangen.de`

using suitable clipping objects (section 7) is introduced. Finally, section 8 presents several results achieved with technical and medical image data demonstrating the value of our approach.

2 Line Integral Convolution

In an early texture–like method, introduced by van Wijk [31] in 1991, oval spots with white noise are distorted along a straight line segment oriented parallel to the local vector direction. LIC itself was introduced by Cabral and Leedom [4] in 1993 who presented an algorithm which performed the convolution along curved stream line segments. In 1995 Stalling and Hege [29] made LIC much faster, more accurate and independent of resolution. Due to these improvements LIC turned out to be very suitable for displaying vector fields on two–dimensional surfaces and became very popular. Hence, a vast quantity of different algorithms and improvements have been developed in the last years. In 1994 Forssell [10] presented an extension that allows to map flat LIC images onto curvilinear surfaces in three dimensions. A problem of this method is the distortion of length during the mapping process. In 1997 Teitzel et al. [30] solved this problem by computing LIC images directly on triangulated surfaces in three–dimensional space without mapping. Another method for creating LIC images on surfaces in three–dimensional space was presented by Mao et al. [21] using solid texturing. Wegenkittl et al. [32] introduced oriented LIC in order to visualize the orientation of the flow and Risquet [25] presented a drastic simplification for accelerating the imaging process. Many other authors have been working on enhancements by color coding [26] or animating LIC [2, 11, 16], by accelerating the image generation [1, 35], or by developing specially adapted techniques for applying LIC to unsteady flows [27].

The LIC algorithm filters an input volume along stream lines, also denoted integral curves or flow lines, of a given vector field and generates a 3D–texture as output. In most cases in scientific visualization a texture with white noise is used as input. The intensity I of an output texture voxel located at $x_0 = \sigma(s_0)$ is given by

$$I(x_0) = \int\limits_{s_0-L}^{s_0+L} k(s-s_0)\, T(\sigma(s))\, ds\,,$$

where $\sigma(s)$ denotes a stream line of the vector field parameterized by arc length, T the intensity of the input texture and k a filter kernel. If we choose a constant filter kernel k and consider that T is constant at each voxel, the convolution integral can be computed by sampling the input texture T at locations x_i along the stream line $\sigma(s)$:

$$I(x_0) = k \sum_{i=-n}^{n} T(x_i)\,,$$

where we choose $k = 1/(2n+1)$ to normalize the intensity. The convolution causes voxel intensities to be highly correlated along individual stream lines but independent in directions perpendicular to them. In the resulting images the stream lines are clearly visible.

As a drawback the original LIC algorithm does not provide information about the absolute value of the velocity. In consequence, several approaches were developed to portray the flow direction and velocity as well as other scalar values. This is accomplished by using color coding [2, 26, 29], asymmetric filter kernels [13, 29] and a varying line width [15, 33]. In order to enhance the insight into LIC volumes, techniques like volume rendering seem to be applicable since the LIC method transforms a 3D vector field into a 3D scalar field of gray values. In this context the animation of a static LIC volume allows to integrate velocity information, if the frame rate is sufficiently high.

3 Direct Volume Rendering with 3D–Texture Mapping

For the visualization of 3D scalar fields, direct volume rendering proved to be very suitable. According to [18], all known approaches of direct volume rendering can be reduced to the transport theory model which describes the propagation of light in materials. Approximations of the underlying equation of light transfer have been developed. They differ considerably in the physical phenomena they account for and in the way the emerging numerical models are solved. The expensive basic ray casting approach [20] has been accelerated in various ways. Among other strategies this comprises adaptive sampling [8, 7], exploiting coherence [19], using special purpose architectures [17, 23] and taking advantage of hardware in graphics workstations [3].

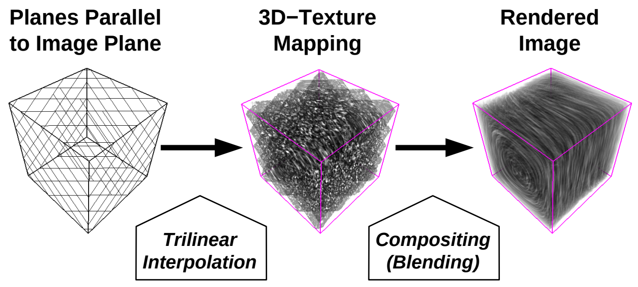

Figure 1: Volume rendering with 3D–texture mapping.

The basic idea of the 3D–texture mapping approach is to use the scalar field as a 3D–texture. At the core of the algorithm, multiple equidistant planes (*slices*) parallel to the image plane are clipped against the bounding box of the volume (see Figure 1). During rasterization hardware is exploited to interpolate 3D–texture coordinates at the polygon vertices and to reconstruct the texture samples by trilinearly interpolating within the volume. Finally, the 3D–representation is produced by successive blending of the textured polygons back–to–front onto the viewing plane. Since this process uses the blending and interpolation capabilities of the underlying hardware, the time consumed for the generation of an image is negligible compared to software based approaches.

As an advantage of this approach, interactive frame rates are achieved even if applied to scalar fields of high resolution. This is an important prerequisite for the comprehensive analysis of volume information. Due to the applied trilinear interpolation scheme and the number of sampling points, which are appropriately adjusted, the resulting images are of high quality. This guarantees to reproduce fine structures and ensures their clear delineation, especially if they are zoomed closely for a detailed inspection.

4 Interactive Assignment of Color and Opacity Values

The graphics hardware is also exploited to modify the texture lookup tables used for the assignment of color and opacity values. This is an indispensable feature in order to enhance or suppress portions of data specified by certain scalar values. The interactive manipulation of the respective transfer functions and the direct update of the 3D–representation considerably simplify the process of finding an appropriate setting. Using predefined lookup tables, which are adjusted by a few manipulation operations, ensures to quickly produce a meaningful visualization of a LIC volume. Thereby, it is easy to obtain a more transparent visualization, which allows to see

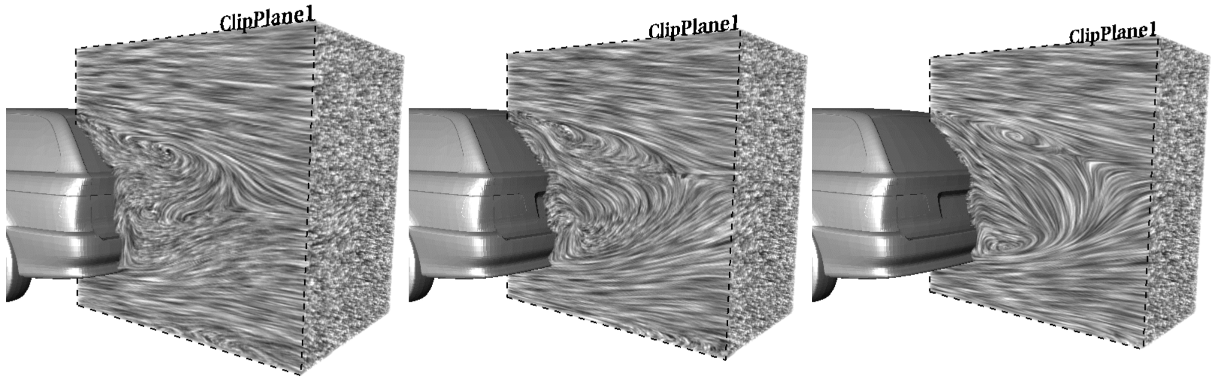

Figure 2: Visualization of a LIC volume: The flow field is explored using a clip plane, which is interactively translated.

an underlying geometry (see Figures 4 and 11). In the same way, complementary information is visually integrated for better orientation if fusion with another volume is performed. As demonstrated in Figure 15 a 3D–LIC calculation within the aorta is combined with the surrounding anatomy. In contrast, the fully opaque assignment shows the flow information directly at the outer surface of the vector field. According to Figure 2 this is useful if a clip plane is applied in order to explore the LIC volume.

In Figure 3 the setting of the transfer functions for color and opacity values is shown which leads to the visualization presented in Figure 4. Although arbitrary transfer functions are applicable a piecewise linear mapping is sufficient. The arrows indicate the location and the direction of simple manipulation operations which are required to adjust the lookup tables. As an additional orientation the intensity histogram of the volume data is displayed within the diagram. If an opaque representation is envisaged *(left side)*, opacity is set to a constant high value. However, it is useful to decrease it slightly in order to improve the visual continuity and impression. Thereby, stream lines become visible which are directly below the actual surface. Simultaneously, a linear ramp is specified for the luminance values enhancing the contrast of the resulting image. Within the histogram this ramp is positioned in the center of the main peak.

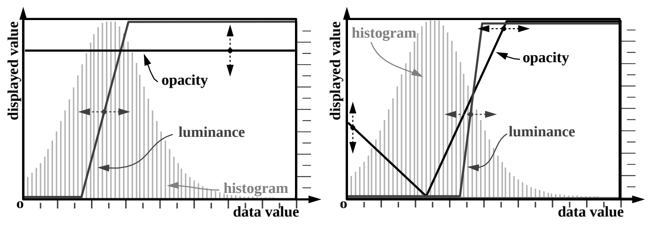

Figure 3: Intensity histogram and transfer functions for the visualization of the LIC volume shown in Figure 4: Setting for the opaque representation *(left)* — Setting for the semi-transparent representation *(right)*.

The semi–transparent representation *(right side)* requires to use low opacity values for low data values and high opacity values for high data values. Further on, a linear ramp of high gradient is used in between in order to produce a smooth transition. Depending on the selected background color, the contrast is intensified if there is another linear ramp for opacity values that increases to lower data values. This is of importance if light background colors are chosen. The transfer function for luminance values is positioned within the transition from low to high opacity values. This leads to a good impression of depth, as can be seen on the right side of Figure 4. Moreover, the interactive adjustment of transfer functions is an efficient way to substitute the separate application of sparse noise textures as proposed in [15].

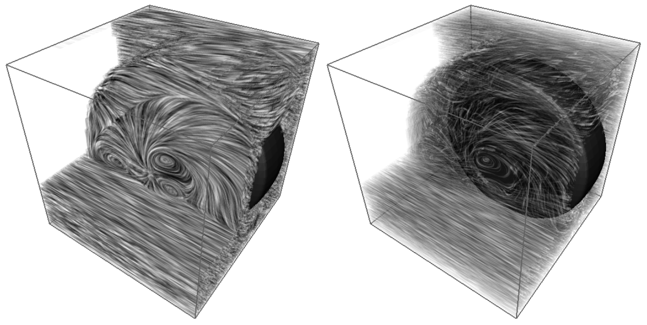

Figure 4: Simulated flow around wheel with different setting of transfer functions: *(left)* Opaque representation showing details at the surface and *(right)* semi–transparent representation efficiently substituting the application of sparse noise textures.

5 Clipping Functionality

Additional scalar fields such as density, pressure, or absolute value of velocity are frequently used in order to specify a volume of interest (VOI) to restrict the rendering process to significant parts of the flow. This VOI is usually applied a priori to the input texture or as a postprocess to the resulting 3D–LIC texture. Since this operation modifies the voxel data, it is impossible to change the VOI during the visualization process.

The above mentioned strategy aims at a visualization of the LIC volume as an opaque object extracted by the VOI. Due to the intricate and dense structure of stream lines inside a 3D–LIC texture, higher transparency will result in cluttered displays. In order to explore the interior structures, the use of clip planes is a straight forward approach. However, for a static visualization clip planes are not sufficient when visualizing 3D–LIC, because planar surfaces do not generally follow the direction of the flow, resulting in discontinuous stream lines. However, the interactivity provided by 3D–texture based volume rendering greatly improves the understanding of interior structures. Using a clip plane in motion, the user is very well capable of tracking lines or surface–shaped structures within the 3D–LIC texture. Figure 2 demonstrates the contribution of interactive clip planes to significantly improve the spatial understanding of the complex turbulent flow at the rear of a car.

Original LIC textures do not contain information about the absolute value of velocity and the sign of the flow direction. To remove this deficiency, the approaches mentioned in section 2 encode the missing information by the use of different shape and color. However, the understanding of these representations is extremely difficult in 3D. Point icons such as vector plots are a straight forward approach to depict both direction and absolute value of velocity, although in 3D they generate cluttered displays. Since drawing arrows leads to good results in 2D, the idea is to restrict the arrow plot to another kind of clip plane in 3D. This plane can be placed and moved in real–time. Then, vectors are drawn only at discrete integer coordinates within the 3D flow field domain to avoid trilinear interpolation of the vector data. To determine these vector positions a standard 3D scan conversion of the specified plane is computed. The density of arrows can additionally be adjusted in order not to produce images overloaded with information. The impression of flow is further enhanced by animating the vector length linear within a specified range. This vector plane can be either placed inside a semi–transparent 3D–LIC texture or attached to one of the clip planes for the volume (see Figure 14).

In addition to the described functionality, an approach for interactive clipping with arbitrary geometry is outlined in section 7. Using this method, it is possible to suppress either the interior or the exterior volume. The ability to use every closed triangle surface as clipping geometry allows the application of fast visualization and animation techniques of high flexibility. To specify a VOI for the LIC texture, the boundary of the clip object should roughly follow the course of the stream lines. Therefore, a straight forward approach is to compute stream surfaces, that form a solid object. This is a fast alternative to the computation of LIC on stream surfaces. In contrast to this, when using time–dependent clipping objects for animation, stream lines should intersect the boundary surface of the clipping geometry orthogonally, unlike the VOI. For this purpose it is obvious to use time surfaces or time volumes as described in the following section.

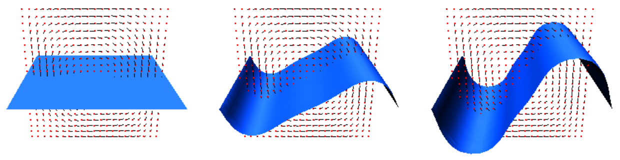

Figure 5: Surfaces of equal time inside a simple cavity flow field.

6 Animating LIC with Time–Dependent Color Tables

To obtain a set of time–dependent clipping objects, an appropriate triangle surface is placed inside the flow field in such a way that stream lines intersect the surface at an angle of preferably 90 degrees. This initial surface is evolved through time by computing particle traces for each vertex of the surface. The result is a set of surfaces of equal time as shown in Figure 5.

To allow for divergent flow regions that cause surface triangles to grow fast in size, a simple subdivision scheme (see Figure 6) is applied to split edges whose length exceed a specified threshold limit. Whenever a triangle is found that has at least one oversized edge, the neighboring triangles are checked to decide whether subdivision can be avoided or not. If the neighboring triangle is small enough, in certain cases the oversized edge can be eliminated by swapping the edge instead of subdividing it. For better performance of the subsequent steps an additional polygon reduction algorithm [5] is applied afterwards to the complete set of surfaces to minimize the total number of triangles. The time consumed for this pre–processing is negligible in comparison to the calculation of 3D–LIC.

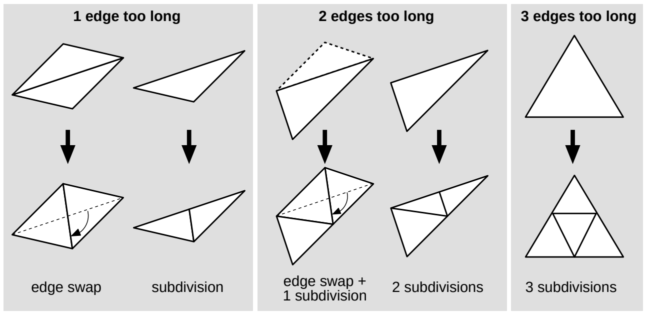

Figure 6: Subdivision of triangles whose edge length exceeds a specified threshold limit.

Subsequently, this pre–computed set of time surfaces is used to divide the LIC volume into disjoint sub–volumes that are located between two adjacent time surfaces. These subsets are numbered consecutively and the index is assigned to every voxel of the subset. The result of this computation is a second volume of the same size as the 3D–LIC that contains the subset indices of the voxels. Note that if time surfaces are intersecting each other, the assignment may be ambiguous, i.e. the subsets may not be disjoint. In this case priorities for the subsets must be specified.

Using the texture–based volume renderer, the subset indices can be used to assign a unique color to every subset when rendering the 3D–LIC texture. Thus, the LIC volume can be animated by sequentially shifting an appropriate color lookup table through the volume subsets. As mentioned above, high–end graphic workstations provide hardware accelerated texture color tables of a fixed size (usually 8 bit). The available color table depth must be split into fixed numbers of bits for the subset index and for the LIC color index. According to this decision, the voxel values of the original LIC volume must be reduced to the appropriate color index depth. Furthermore, a modulo function must be applied to the subset indices if the number of subsets exceeds the available range. Finally, the reduced LIC volume and the volume containing the subset indices are merged together.

For the example data animated LIC with 8 bit lookup tables was used. A subset index depth of 3 bit was chosen, which allows to display 8 different subsets each with 32 different colors (see Figure 7 *top*). Figure 13 shows animation frames of a cavity flow field. The color table is used to move regions of different colors through the 3D–LIC volume with equal opacity curves for each subset (see Figure 7 *middle*). The animation is done by shifting the color indices sequentially by 32. Another animation technique is to keep the color curve constant and to shift a varying opacity value through the subsets (see Figure 7 *bottom*). This causes the different subsets to appear and disappear through time. Alternatively, color tables with 4 bit for subset indices can be used to display 16 different subsets each with a color index range of 16.

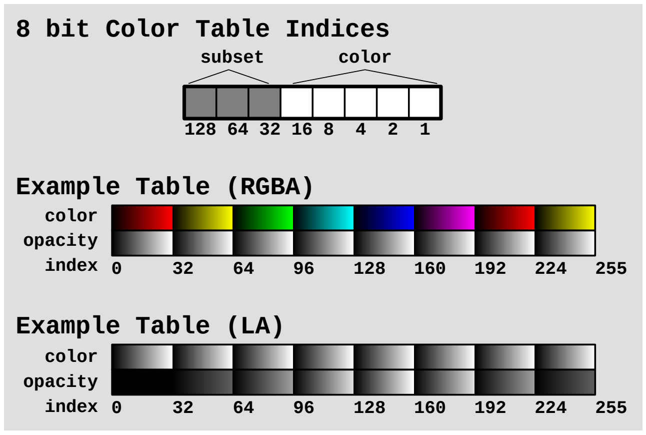
8 bit Color Table Indices
subset
color
128 64 32 16 8 4 2 1
Example Table (RGBA)
color
opacity
index 0 32 64 96 128 160 192 224 255
Example Table (LA)
color
opacity
index 0 32 64 96 128 160 192 224 255

Figure 7: Look–up tables for color animation: *(top)* assignment of available bits to access 8 subsets each with 32 color and opacity values, *(middle)* RGBA–table shifting color values, *(bottom)* luminance–alpha (LA)–table shifting opacity values.

7 Animating LIC with Arbitrary Clipping Geometries

Westermann [34] introduced a new method that combines 3D–texture based volume rendering with a per–pixel frame buffer locking mechanism to clip the volume against arbitrary geometries. As long as the object is a closed triangle surface with definite vertex ordering, it can efficiently be used as clipping geometry.

The basic idea is to determine all pixels that are covered by the cross–section between the object and the current slicing plane when rendering the volume. Then, in order to prevent the textured polygon from getting drawn to these locations, the pixels are locked. To implement this frame buffer locking mechanism, the stencil test provided by OpenGL is used. When writing pixels to a frame buffer position during rasterization, the contents of the stencil buffer at the corresponding position is compared to a user specified reference value to decide whether the frame buffer write should be accepted or rejected. To efficiently determine whether a pixel is covered by the cross–section or not, the clipping geometry is rendered in polygon mode directly into the stencil buffer, leaving the frame buffer untouched. Afterwards the textured polygons can be drawn into the frame buffer using the stencil buffer contents as pixel mask.

This technique can be used to interactively clip the 3D–LIC texture against a pre–computed set of time volumes. Since the clipping object for this method must be solid, the computation of time surfaces used in the previous section must be adapted to produce closed surfaces. Generally, there are two ways to obtain such closed clipping objects. On the one hand you can use a closed surface as initial surface for time surface computation and take care that the topology is not corrupted, e.g. by high vorticity or vertices that leave the flow field boundaries. On the other hand you can compute arbitrary 2D time surfaces as in the previous section and use them to construct volume objects afterwards, for instance by joining two adjacent time surfaces to form a closed object. Figure 8 shows volume objects which are computed by combining 2D time surfaces with the bounding box of the flow field. Using these clipping geometries, the 3D–LIC texture is animated by sequentially switching between different time volumes.

The cavity flow field (see Figure 9) was animated by closed clipping volumes constructed from initially parallel time surfaces. The flow field is the same as the one used for color–animation in Figure 13. In Figure 10 a flat box was placed into the a turbulent flow field and evolved through time, resulting in rather complex time volumes.

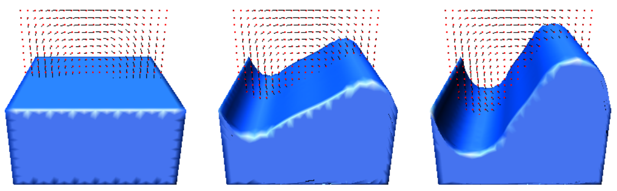

Figure 8: Closed surfaces built by combining the time surfaces of Figure 5 with the flow field bounding box. These objects can effectively be used as clipping geometry.

8 Results

The availability of the presented approaches strongly depends on the capabilities of the underlying hardware. Throughout our experiments we used a SGI Octane MXE with 4 MB of texture memory and a SGI Onyx2 (R10000, 195MHz) with BaseReality graphics hardware providing 64 MB of texture memory. The frame rates are comparable on both architectures with the Onyx2 giving higher performance for bigger 3D–textures since no bricking is required. However, the limiting factor for clipping with arbitrary geometry is access of the stencil buffer.

The presented algorithms were integrated into an interactive volume viewer based on OpenInventor presented by Hastreiter [12] and Sommer [28], which provides a standardized user interface. The basic functionality of OpenInventor, including different 3D–viewers and a variety of object manipulators, enables intuitive handling and precise placement of clipping planes. This is indispensable for the meaningful visualization of scientific vector data.

Figure 11 shows the visualization of a simulated velocity field within a wheel casing of a car. On the left side the entire and fully opaque LIC volume is presented, giving an optimal impression of the situation at the boundary surface between object and and flow field. In order to view interior structures a clip plane is translated through the volume as can be seen in the middle image. Note that the spatial understanding is conveyed by motion and is hard to be obtained from a still image. The semi–transparent volume is revealing parts of the wheel geometry. Interactive adjustment of transfer functions allows to produce a variety of different representations of the data, using a single LIC texture. This is a faster and more flexible approach in comparison to the computation of LIC with multiple sparse noise textures of varying density. For the image on the right a modified lookup table for color and opacity is applied to the full 3D–LIC texture.

Interactive clipping using frame buffer locks allows the application of arbitrary clipping geometries. Figure 12 presents frames of an animation sequence of the same velocity field, generated within a user–defined region of interest. The successive sub–volumes were computed with the approach, presented in section 7, using a semi-circular object as initial time surface. Compared to the static 3D–representation, the animation sequence significantly enhances the understanding of air flow around the axle and the resulting vorticity inside the wheel casing.

As for each volume slice the clipping geometry must be traversed twice, objects with a large number of triangles will noticeably degrade performance (see Table 1). To achieve interactive frame rates in most cases it is vital to decrease the number of triangles using some polygon reduction algorithm.

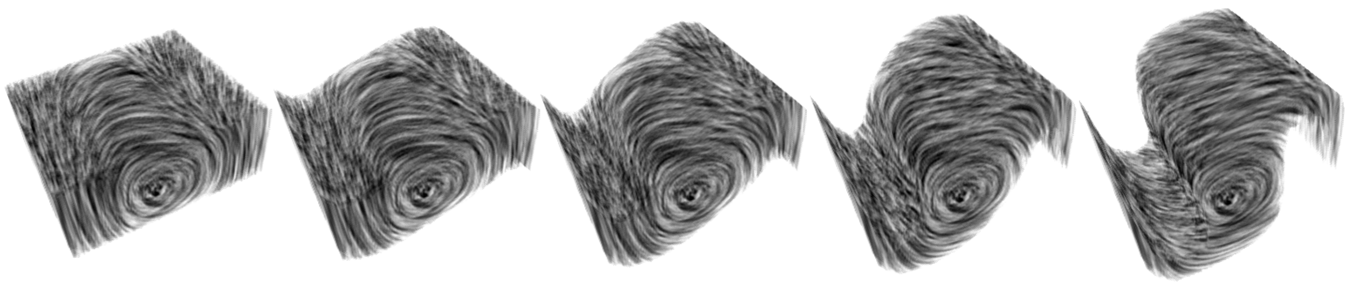

Figure 9: Cavity flow field visualized by clipped 3D–LIC.

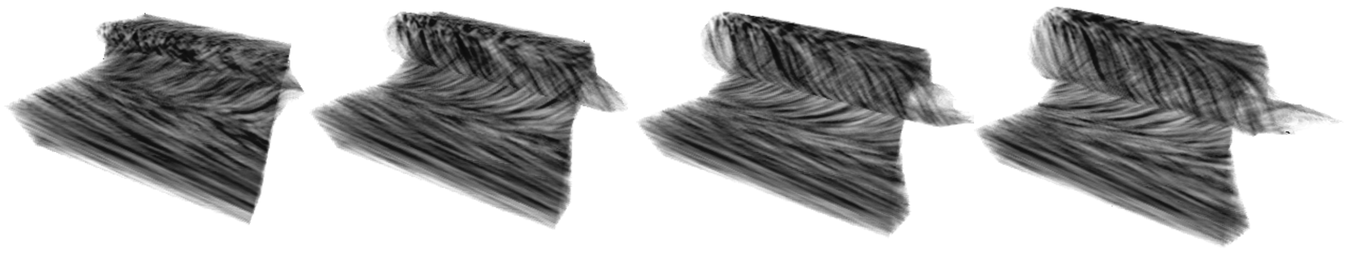

Figure 10: Pegase data set visualized by clipped 3D–LIC

performance of color–animated LIC

data set	LIC resolution	frames per second
cavity (Fig. 13)	128^3	16–20

performance of clipped LIC

data set	LIC resolution	triangles per frame	frames per second
Cavity (Fig. 9)	128^3	269	3–4
Pegase (Fig. 10)	$128^2 \times 256$	653	2–3
Wheel (Fig. 12)	256^3	4367	< 1

Table 1: Performance of displaying animated 3D–LIC on a SGI Octane MXE with a window size of 640×400 pixel

While using color table animation as presented in section 6, the maximum number of gray values or color indices is limited. However, noise textures, that are used as input for LIC algorithms, usually are generated by random functions which can easily be adapted to a reduced range of values. From experience the use of sparse noise textures as suggested in [15] will also result in LIC images with a limited number of gray values. Figure 13 demonstrates the application of animated LIC with time-dependent transfer functions in case of a cavity flow field. Using hardware–accelerated texture color tables the animation sequence can be displayed with high frame rates (see Table 1).

As mentioned before, intersecting time surfaces might lead to voxel subsets that are not disjoint. Although the problem can be solved by specification of priorities, the resulting animation in such ambiguous regions is incorrect. In some cases this can be avoided by intelligently placing the initial time surface. In general however, the approach based on time-dependent color tables is limited to less intricate flow fields.

Figure 14 demonstrates the flow situation at the rear side of a car. A clip plane is used to reveal the interior turbulence. Color–encoded vector plots are integrated on the plane, adding information about direction and absolute value of velocity. With this representation it is easy to identify regions of the flow with opposite directions that cause vorticity. Again, interactive placement of the planes is vital for a comprehensive analysis.

Figure 15 shows the application of interactive flow visualization for an improved medical diagnosis of blood vessel malformations. In this case the analysis is based on vector data acquired with 3D–phase contrast angiography *(PCA)*. PCA is a specific pulse sequence used for magnetic resonance imaging, which is not yet commonly established in medical routine. This is related to long acquisition times and still limited spatial resolution. However, as this technique evolves, it will significantly contribute to improve the diagnosis of aneurysms and stenosis.

The left image shows the aorta, the big artery in front of the spinal column. A vector plot of the original PCA data set is added to the clipping plane within the anatomical volume data. Different time steps are used to animate the arrow plot. In order to compute a 3D–LIC volume, preprocessing such as noise reduction is required to remove artifacts and enhance noisy data. Subsequently, standard magnetic resonance angiography *(MRA)* data showing the vessel anatomy at high resolution is used to extract the vascular structures. This segmentation is used as voxel mask for the vector field before computing the LIC texture. Integration of LIC data into the original anatomical image data as displayed in the middle and the left image, ensures good anatomical orientation.

9 Conclusion

In this paper we have introduced interactive direct volume rendering as means for efficiently visualizing 3D–LIC. Volume rendering based on 3D–texture mapping proved to be ideal since it provides the necessary interactivity and image quality. In the same context the application of transfer functions which are intuitively manipulated are an indispensable prerequisite for a convenient and comprehensive exploration of LIC volumes. In order to improve the information content of the 3D–visualization vector plots are optionally combined with clip planes. Thereby, the absolute value and the orientation of the velocity is effectively integrated.

We have presented two approaches to display animated 3D–LIC by using time surfaces for clipping. The animation was done on the one hand by computing a special 3D–LIC texture for the use with color table animation and on the other hand by clipping the 3D–volume interactively against pre–computed time volumes.

Our results have shown that animation of 3D–LIC greatly improves the perception of flow in 3D, and can be done interactively without the use of pre–computed 3D–textures. Furthermore, we have shown that high frame rates can be achieved on modern high–end graphics workstations by exploiting hardware features such as texture color tables or frame buffer locks.

Efficient computation of time surfaces and automatic placement of the initial surface, as well as the adaptation of the presented approaches to medical flow data such as 3D–phase contrast angiography are topics of our future work.

10 Acknowledgments

We are grateful to BMW for providing the car related flow simulation. Additionally, we would like to thank Dave Parker from the Department of Mechanical Engineering at Stanford University for the 3D–PCA data. Finally, we also thank Sven Enger from the Department of Fluid Mechanics of the University of Erlangen for making the cavity data set available.

References

[1] H. Battke, D. Stalling, and H.-C. Hege. Fast Line Integral Convolution for Arbitrary Surfaces in 3D. In H.-C. Hege and K. Polthier, editors, *Visualization and Mathematics*, pages 181–195. Springer Verlag, 1997.

[2] J. Becker and M. Rumpf. Visualization of time-dependent velocity fields by Texture Transport. In *Proc. Visualization in Scientific Computing '98*, pages 91–101. Springer-Verlag, 1998. Eurographics Workshop in Blaubeuren, Germany.

[3] B. Cabral, N. Cam, and J. Foran. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware. *ACM Symp. on Vol. Vis.*, pages 91–98, 1994.

[4] B. Cabral and L. Leedom. Imaging Vector Fields Using Line Integral Convolution. In *Proc. of SIGGRAPH*, Comp. Graph. Conf. Series, pages 263–270, 1993.

[5] S. Campagna, L. Kobbelt, and H.-P. Seidel. Enhancing Digital Documents by Including 3D-Models, 1998. To appear in Comp.s & Graphics.

[6] R. Crawfis and N. Max. Texture Splats for 3D Scalar and Vector Filed Visualization. In *Proc. Visualization*, pages 261–266. IEEE Comp. Soc. Press, 1993.

[7] J. Danskin and P. Hanrahan. Fast Algorithms for Volume Ray Tracing. In *Worksh. of Vol. Vis.*, pages 91–98. ACM, 1992.

[8] B. Drebin, L. Carpenter, and P. Hanrahan. Volume Rendering . *Comp. Graphics*, 22(4):65–74, 1988.

[9] D. Ebert, R. Yagel, J. Scott, and Y. Kurzion. Volume Rendering Methods for Computational Fluid Dynamics Visualization. In *Proc. Visualization*, pages 232–239. IEEE Comp. Soc. Press, 1994.

[10] L. Forssell. Visualizing Flow Over Curvilinear Grid Surfaces Using Line Integral Convolution. In *Proc. Visualization*, pages 240–247. IEEE Comp. Soc. Press, 1994.

[11] L. Forssell and S. Cohen. Using Line Integral Convolution for Flow Visualization: Curvilinear Grids, Variable-Speed Animation and Unsteady Flows. *IEEE Trans. on Vis. and Comp. Graph.*, 1(2):133–141, 1995.

[12] P. Hastreiter, H. Çakmak, and T. Ertl. Intuitive and Interactive Manipulation of 3D Data Sets by Integrating Texture Mapping Based Volume Rendering into the OpenInventor Class Hierarchy. In *Worksh. Bildverarb. f.d. Med. (BVM)*, pages 149–154, 1996.

[13] H.-C. Hege and D. Stalling. Fast LIC with Piecewise Polynomial Filter Kernels. In H.-C. Hege and K. Polthier, editors, *Mathematical Visualization*, pages 295–314. Springer-Verlag, 1998.

[14] J. Hultquist. Interactive numerical flow visualization using stream surfaces. In *Computing Systems in Engineering*, pages 349–353, 1990.

[15] V. Interrante and C. Grosch. Visualizing 3D Flow. *IEEE Comp. Graph. and Appl.*, 18(4):47–53, 1998.

[16] B. Jobard and W. Lefer. The Motion Map: Efficient Computation of Steady Flow Animations. In *Proc. Visualization*, pages 323–328. IEEE Comp. Soc. Press, 1997.

[17] G. Knittel and W. Straßer. A Compact Volume Rendering Accelerator. In A. Kaufman and W. Krüger, editors, *Symp. on Vol. Vis.*, pages 67–74. ACM SIGGRAPH, 1994.

[18] W. Krüger. The Application of Transport Theory to the Visualization of 3D Scalar Data Fields. In *Proc. Visualization*, pages 273–280, Los Alamitos, CA, 1990. IEEE Comp. Soc. Press.

[19] P. Lacroute and M. Levoy. Fast Volume Rendering Using a Shear–Warp Factorization of the Viewing Transform . *Comp. Graphics*, 28(4):451–458, 1994.

[20] M. Levoy. Display of Surfaces from Volume Data. *IEEE Comp. Graph. and Appl.*, 8(3):29–37, 1988.

[21] X. Mao, M. Kikukawa, N. Fujita, and A. Imamiya. Line Integral Convolution for 3D Surfaces. In *Visualization in Scientific Computing '97*, pages 57–69. Springer Verlag, 1997. Proc. of Eurographics Workshop in Boulogne-sur-Mer, France.

[22] N. Max, B. Becker, and R. Crawfis. Flow volumes for interactive vector field visualization. In *Proc. Visualization*, pages 19–24. IEEE Comp. Soc. Press, 1993.

[23] H. Pfister and A. Kaufman. Cube–4 — A Scalable Architecture for Real–Time Volume Rendering. In R. Crawfis and C. Hansen, editors, *Symp. on Vol. Vis.*, pages 47–54. ACM SIGGRAPH, 1996.

[24] F.J. Post, T. van Walsum, and F.H. Post. Iconic Techniques for Feature Visualization. In *Proc. Visualization*, pages 288–295. IEEE Comp. Soc. Press, 1995.

[25] C. Risquet. Visualizing 2D Flows: Integrate and Draw. In *Eurogr. Workshop on Vis. in Sc. Comp.*, pages 132–142, Blaubeuren, Germany, 1998. The EuroGraphics Ass.

[26] H.-W. Shen, C. Johnson, and K. Ma. Visualizing Vector Fields Using Line Integral Convolution and Dye Advection. In *Symp. on Volume Visualization*, pages 63–70. ACM SIGGRAPH, 1994.

[27] H.-W. Shen and D. Kao. UFLIC Line Integral Convolution Algorithm for Visualizing Unsteady Flows. In *Proc. Visualization*, pages 317–322. IEEE Comp. Soc. Press, 1997.

[28] O. Sommer, A. Dietz, R. Westermann, and T. Ertl. An Interactive Visaulization and Navigation Tool for Medical Volume Data. In V. Skala, editor, *WSCG'98 Conf. Proc.*, Feb. 1998.

[29] D. Stalling and H.-C. Hege. Fast and Resolution Independent Line Integral Convolution. In *Proc. of SIGGRAPH*, Comp. Graph. Conf. Series, pages 249–256, 1995.

[30] C. Teitzel, R. Grosso, and T. Ertl. Line Integral Convolution on Triangulated Surfaces. In *Int. Conf. in Central Europe on Comp. Graph. and Vis. (WSCG)*, volume III, pages 572–581, Plzen, Czech Republic, 1997. Univ. of West Bohemia Press.

[31] J. van Wijk. Spot Noise — Texture Synthesis for Data Visualization. In *Comp. Graph. Proc.*, volume 25 of *Ann. Conf. Series*, pages 309–318. ACM SIGGRAPH, Addison-Wesley, 1991.

[32] R. Wegenkittl and E. Gröller. Fast Oriented Line Integral Convolution for Vector Field Visualization via the Internet. Technical Report TR-186-2-97-07, Vienna University of Technology, Institute of Computer Graphics, Visualisation and Animation Group, 1997.

[33] R. Wegenkittl and E. Gröller. Fast Oriented Line Integral Convolution for Vector Field Visualization via the Internet. In *Proc. Visualization*, pages 309–316. IEEE Comp. Soc. Press, 1997.

[34] R. Westermann and T. Ertl. Efficiently Using Graphics Hardware in Volume Rendering Applications. In *Proc. of SIGGRAPH*, Comp. Graph. Conf. Series, pages 169–177, 1998.

[35] M. Zöckler, D. Stalling, and H.-C. Hege. Parallel Line Integral Convolution. *Parallel Comp.*, 23(7):975–989, 1997.

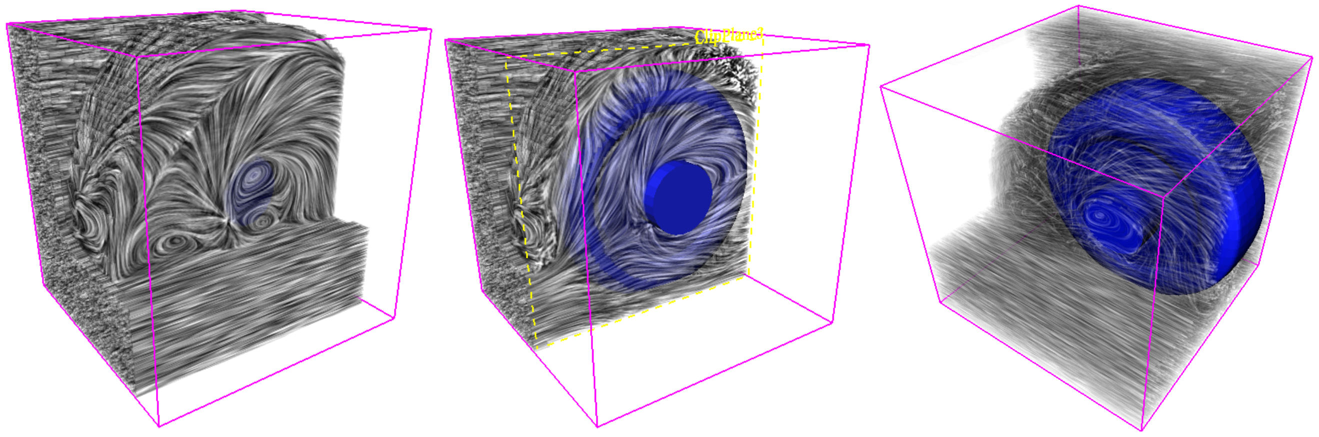

Figure 11: Visualization of LIC showing a simulation of flow around a wheel.

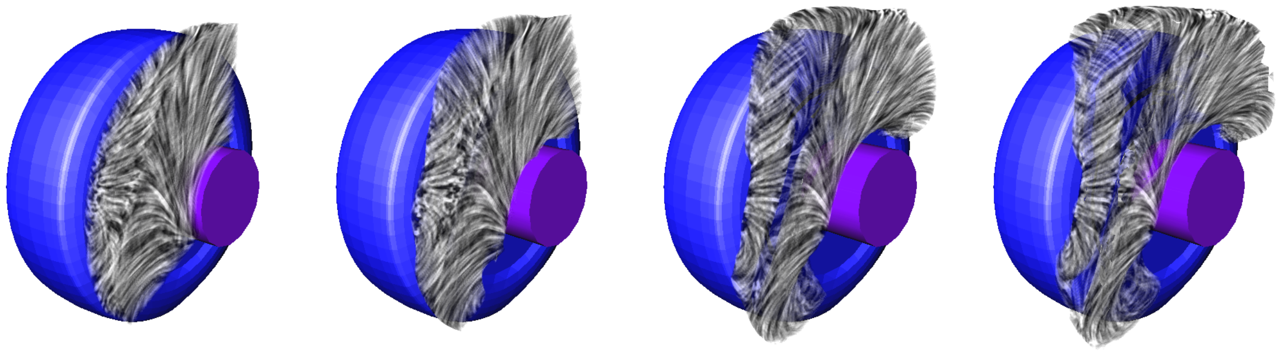

Figure 12: Animation of LIC showing a simulation of flow around a wheel.

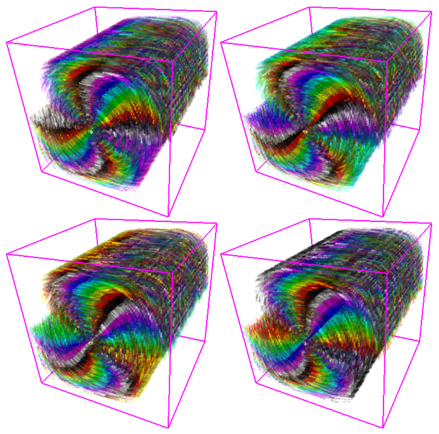

Figure 13: Color animated 3D–LIC.

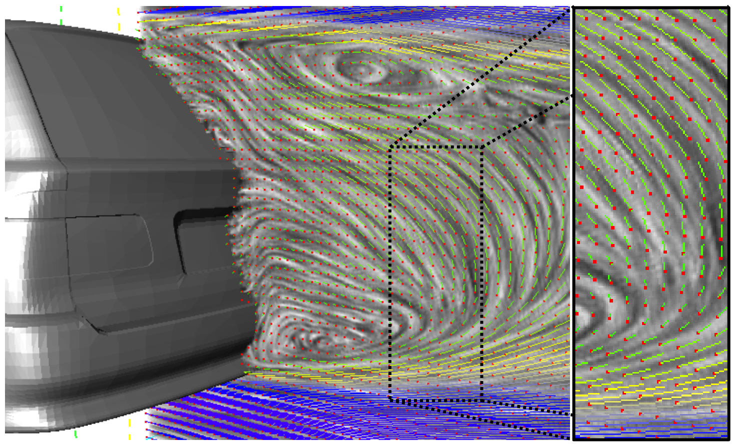

Figure 14: Visualization of LIC combined with an arrow plot.

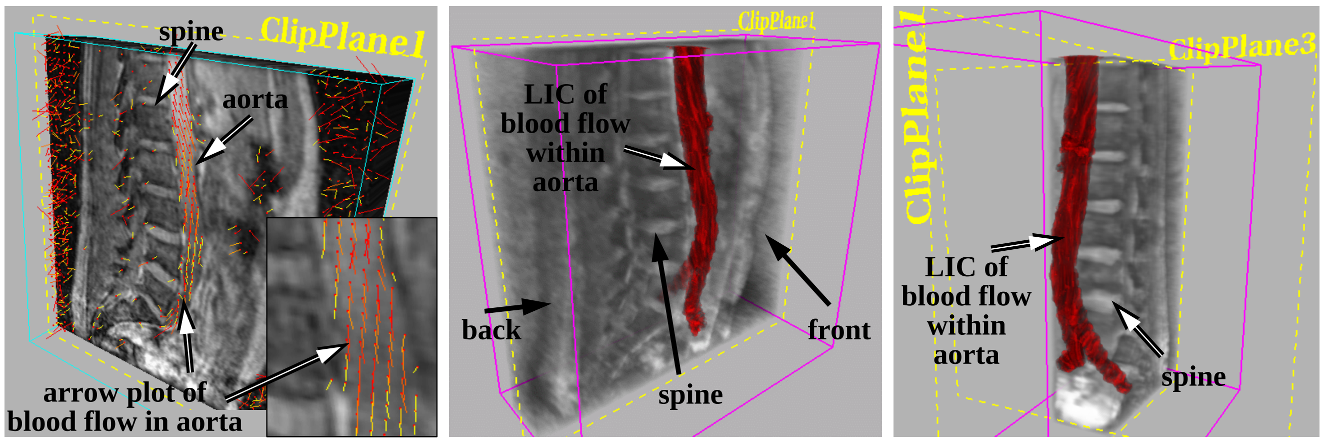

Figure 15: Fusion of anatomical information and 3D–LIC within the aorta based on phase contrast angiography data.