Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2004), pp. 1–10
A. Chairpersona, B. Chairpersonb (Editors)

Texture-Based Flow Visualization on Isosurfaces

Robert S. Laramee,[1] Jürgen Schneider,[2] and Helwig Hauser[1]

[1]VRVis Research Center, Vienna, Austria, {Laramee,Hauser}@VRVis.at, http://www.VRVis.at
[2]AVL, Graz, Austria, Juergen.Schneider@avl.com, http://www.avl.com

Abstract

Isosurfacing, by itself, is a common visualization technique for investigating 3D vector fields. Applying texture-based flow visualization techniques to isosurfaces provides engineers with even more insight into the characteristics of 3D vector fields. We apply a method for producing dense, texture-based representations of flow on isosurfaces. It combines two well know scientific visualization techniques, namely iso-surfacing and texture-based flow visualization, into a useful hybrid approach. The method is fast and can generate dense representations of flow on isosurfaces with high spatio-temporal correlation at 60 frames per second. The method is applied in the context of CFD simulation data, namely, the investigation of a common swirl flow pattern and the visualization of blood flow.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.7 [Computer Graphics] Three-Dimensional Graphics and Realism–Color, shading, shadowing, and texture; [Simulation and Modeling]: Simulation Output Analysis

1. Introduction

At the VRVis Research Center we collaborate with AVL in order to provide visualization solutions for analysis of their CFD simulation result data. AVL (www.avl.com) is an internationally recognized leader in providing automotive design and CFD simulation solutions to its partners in the automotive industry. AVL's own engineers as well as engineers at industry affiliates use visualization software to analyze and evaluate the results of their automotive design and simulation.

1.1. The Case of Swirl Flow

For many of the automotive components that undergo evaluation, there is an ideal pattern of flow the engineers are trying to create. Figure 1 illustrates the swirl motion of fluid flow in a combustion chamber from a diesel engine. In order to generate swirl motion, fluid enters the combustion chamber from the intake ports. Later on in the engine cycle, the kinetic energy associated with this swirl motion is used to generate turbulence for mixing of fresh oxygen into the fluid. The more turbulence generated, the better the mixture of air and diesel fuel, and thus the better the combustion itself. Ideally, enough turbulent mixing is generated such that 100% of the fuel is burned.

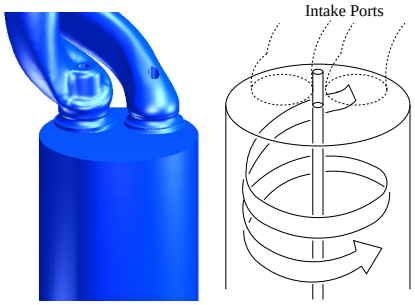

Figure 1: *The swirling motion of flow in the combustion chamber of a diesel engine (side view). The intake ports at the top provide the tangential component of the flow necessary for swirl.*

Since it is the swirling flow that is used to generate turbulence, the swirl should be maximized in order to maximize turbulence. From the point of view of the mechanical engineers designing the intake ports, increased swirl flow leads to some beneficial conditions: (1) improved mixture preparation, .i.e., more fuel contact with oxygen, (2) a higher EGR

(Exhaust Gas Ratio) which means a decrease of fuel consumption, and (3) lower emissions. However, too much swirl displaces the flame used to ignite the fuel. As such, a balance must be achieved between (1) generating enough swirl flow in order to create turbulence and (2) not displacing the flame used to ignite the flow.

Some routine questions that a mechanical engineer may ask when investigating swirl flow are: (1) Can visualization provide insight into or verify the characteristic shape(s) or behavior of the flow? (2) What tool(s) can help to visualize the swirl flow pattern? and (3) Where in the combustion chamber is the swirl flow pattern *not* being met?

1.2. Isosurfaces

Engineers often start an analysis of CFD simulation data using techniques to visualize the flow at the surface in order to get a first impression of the simulation results. The next logical step is to investigate the properties of the flow *inside* the volume. Two-dimensional slices are commonly used, but visualizing 3D characteristics of the flow like swirl can be difficult with 2D slices. Engineers are interested in visualization techniques that provide insight into the spatial dimension orthogonal to the slice as well.

Isosurfaces are a visualization tool used routinely by mechanical engineers to investigate the properties of the flow inside a 3D volume. The shape of an isosurface can give the engineer insight into its 3D characteristics. One reason engineers use isosurfaces, as opposed to say streamsurfaces, is because they are so common. They feel very comfortable with isosurfaces because, like isolines, they are very familiar. We even see isolines in our daily weather reports. The mechanical engineers we spoke to are not as familiar with the notion of a streamsurface and even less its interpretation.

Figure 2 shows an isosurface in the combustion chamber of the data set in Figure 1. The engineer can see that the flow has some of the swirling orientation that they are looking for. However, what is missing from Figure 2 is a clear indication of flow direction, e.g., the upstream and downstream nature of the flow. In particular, it is not obvious where the flow does *not* follow the ideal swirl pattern that the combustion chamber should encapsulate.

1.3. Applying Texture-Based Flow Visualization

Applying texture-based flow visualization techniques to such isosurfaces provides engineers even more insight into the characteristics of 3D vector fields. And this has become a feasible option only recently. We apply the method of Laramee et al. [6] for producing dense, texture-based representations of flow on isosurfaces. The result is a combination of two well know scientific visualization techniques, namely iso-surfacing and texture-based flow visualization, into a useful hybrid approach. Our application is a versatile visualization technique with the following characteristics:

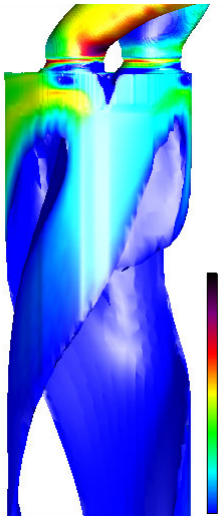

Figure 2: *The swirling motion of flow in the combustion chamber of a diesel engine (side view) as illustrated by an isosurface. This is a velocity isosurface with an isovalue of 5.0 m/s. Any CFD attribute can be mapped to hue.*

- generates a dense representation of flow on adaptive resolution isosurfaces
- visualizes flow on complex isosurfaces composed of polygons whose number is on the order of 200,000 or more
- visualizes flow independent of the isosurface mesh's complexity and resolution
- supports user-interaction such as rotation, translation, and zooming always maintaining a constant, high spatial resolution
- produces fast animations, realizing up to 60 frames per second

The performance is due, among other reasons, to the exploitation of graphics hardware features and utilization of frame-to-frame coherency. We note that the method must be applicable to adaptive resolution isosurfaces like that of Figure 3. Note that many of the polygons in Figure 3 cover less than one pixel. The isosurface algorithm used here is an extension of the Marching Cubes [7] and Marching Tetrahedra [13] algorithms that takes into account more cell types. It handles adaptive resolution meshes in the same spirit as Laramee and Bergeron [5].

The rest of the paper is organized as follows: in Section 2 we discuss related work, Section 3 reviews the research that the work here is built upon. Section 4 details texture-based flow visualization on isosurfaces from CFD. Results and conclusions, including a discussion of the user questions, are presented in Section 5.

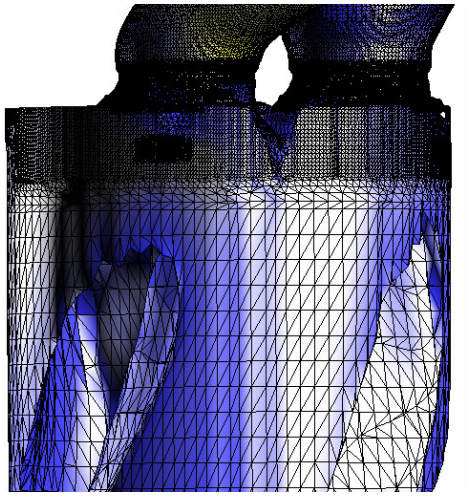

Figure 3: *A close-up, wire-frame view of the isosurface from Figure 2. The algorithm we describe must be applicable to adaptive resolution isosurface meshes.*

2. Related Work

The dense, texture-based category of fluid flow visualization techniques generally started out with Spot Noise [14] and LIC [2]. The main advantage of the texture-based class of algorithms is their *complete* depiction of the flow field while their primary drawback is, in general, the computational time required to produce the results.

2.1. Previous Work with Surfaces

Previous research with a focus on representations of the vector field on boundary surfaces is generally restricted to steady-state flow on simple boundary geometries such as spheres. This is mainly due to the prohibitive computational time required. An enhanced version of Spot Noise is applied to surfaces by De Leeuw and Van Wijk [3].

Some approaches are limited to curvilinear surfaces. Forssell and Cohen [4] extend LIC to curvilinear surfaces with animation techniques and add magnitude and directional information. Stalling [12] provides a helpful overview of LIC techniques applied to surfaces. In particular, a useful comparison of parameterized vs. non-parameterized surfaces is given. However, isosurfaces, like those generated by the Marching Cubes algorithm [7] are not, in general, parameterizable.

Battke et al. [1] describe an extension of LIC for arbitrary surfaces in 3D. The method works by tessellating a given surface with triangles. The triangles are packed (or tiled) into texture memory and a local LIC texture is computed for each triangle. The results are limited to relatively small, (1,600-4,000 polygons) simple surfaces composed of equilateral triangles however. Furthermore, the reported computation times are on the order of one minute.

Mao et al. [8] extend LIC for visualizing a vector field on an arbitrary surface in 3D. The convolution of a 3D white noise image, with filter kernels defined along the local streamlines, is performed only at visible ray-surface intersections. However, ray tracing is, in general, costly. The results presented there required 6 minutes of processing time for a surface mesh composed of 10,000 triangles.

Our review of the literature [9, 10] indicates that only recently have dense, texture-based flow visualization techniques on surfaces become more feasible. Perhaps this is one reason why we have not seen them applied to isosurfaces. Laramee et al. [6] and Van Wijk [15] both present texture-based flow visualization for boundary surfaces. In both methods, texture is advected in the direction of the flow at fast frame rates. Also, both methods are suited for the visualization of unsteady flow on surfaces. The application we present here builds on the work of Laramee et al. [6]. In what follows, we describe why this method is well suited for the case of isosurfaces and discuss how it can be useful.

3. Method Background

In order to understand how and why we apply texture-based flow visualization to isosurfaces, we briefly outline the method background. In brief, the algorithm presented by Laramee et al. [6] simplifies the problem by confining the advection of texture properties to image space. The surface geometry is projected to image space and then a series of textures are mapped, blended, and advected. This order of operations eliminates portions of the surface hidden from the viewer. In short, the previous method for visualization of flow on surfaces is comprised of the procedure depicted schematically in Figures 4 and 5. Each step of the pipeline is necessary for the dynamic cases of changes to the isovalue, time-dependent geometry, rotation, translation, and scaling, and only a subset is needed for the static cases involving no changes to the view-point or isovalue. We consider these stages in briefly in the sections that follow. They are described in more detail in previous research [6].

3.1. Vector Field Projection

We start with a given surface with the 3D velocity data stored at the vertices of the polygons. In order to advect texture properties in image space, the vector field associated with the surface is projected to the image plane. The method here takes advantage of the graphics hardware. A color whose R, G, and B values encode the x, y, and z components of the local vectors is assigned to each vertex of the surface respectively. The velocity-colored geometry is rendered to the framebuffer. The term *velocity image*, Figure 5 top, left, is used to describe the result of encoding the velocity vectors at the mesh vertices into color values. The velocity image is interpreted as the vector field and is used for the texture advection in image space.

During the construction of the velocity image, Gouraud

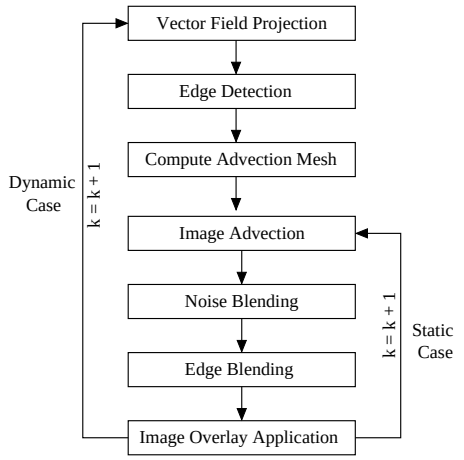

Figure 4: *Flow diagram of texture-based flow visualization on complex surfaces -k represents time as a frame number.*

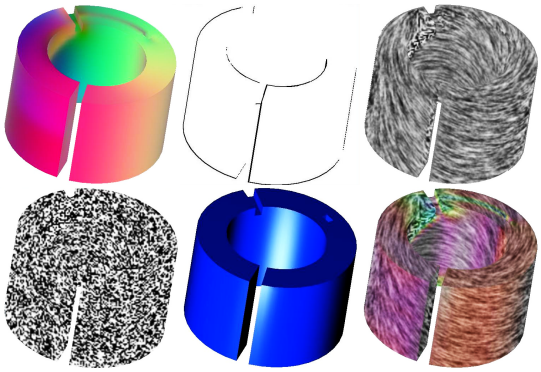

Figure 5: *The 5 conceptual stages, plus a 6^{th} composite image, used for the visualization of surface flow on a ring: (top, left) the velocity image, (top, middle) the geometric edge boundaries, (top, right) the advected and blended textures, (bottom, left) a sample noise image, (bottom, middle) an image overlay, (bottom, right) the result of the composited images with an optional color map. The geometric edge boundaries are drawn in black for illustration.*

Shading, also supported by the graphics hardware, is enabled. Since each velocity component is stored as hue at each polygon vertex of the surface, the smooth interpolation of hue amounts to hardware-assisted vector field reconstruction. The de-coded velocity vectors used to compute the advection mesh (Sec 3.2) are then projected onto the image plane.

3.2. Advection Mesh Computation and Image Advection

After the projection of the vector field the mesh used to advect the textures is computed. A regular, rectilinear mesh is distorted according to the velocity vectors stored at mesh grid intersections. The distorted mesh vertices can then be computed by backward advecting each mesh grid intersection according to a discretized Euler approximation of a pathline, $\mathbf{p}$,

$$\mathbf{p}_{k-1} = \mathbf{p}_k - \mathbf{v}^p(\mathbf{p}_{k-1}; t)\Delta t \tag{1}$$

where $\mathbf{v}^p$ represents a magnitude and direction after projection to the image plane and k represents time as a frame number. In this case the texture coordinates located at the backward distorted mesh positions are mapped to the regular, rectilinear mesh vertices.

3.3. Edge Detection and Blending

While many advantages are gained by decoupling the image advection process from the 3D surface geometry, artifacts can result, especially in the case of geometries with sharp edges. For example a portion of the 3D geometry can be much less visible after the projection onto the image plane and can even look like an edge. If the flow texture properties are advected across such an edge in image space an artificial continuity can result. A geometric edge detection process is incorporated into the algorithm in order to handle this (depicted schematically in Figures 4 and 5 top, middle).

During the image integration computation, we compare spatially adjacent depth values during one integration and advection step. We compare the associated depth values, z_{k-1} and z_k in world space of $\mathbf{p}_{k-1}$ and $\mathbf{p}_k$ from equation 1, respectively. If

$$|z_{k-1} - z_k| > \varepsilon \cdot |\mathbf{p}_{k-1} - \mathbf{p}_k| \tag{2}$$

where ε is a threshold value, then we identify an edge. All positions, $\mathbf{p}$, for which equation 2 is true, are classified as edge crossing start points. Special treatment must be given when advecting texture properties from these points.

3.4. Noise Blending

By reducing the image generation process back to two dimensions, the noise injection and blending phase (Figure 5 top, right) falls in line with the original IBFV technique, [15] namely, an image, F, is related to a previous image, G, by:

$$F(\mathbf{p}; k) = \alpha \sum_{i=0}^{k-1} (1-\alpha)^i G(\mathbf{p}_{k-i}; k-i) \tag{3}$$

where $\mathbf{p}$ represents a pathline, α defines a blending coefficient. Thus a point, $\mathbf{p}_k$, of an image F_k, is the result of a convolution of a series of previous images, $G(\mathbf{x}; i)$, along the pathline through $\mathbf{p}_k$, with an decay filter defined by $\alpha(1-\alpha)^i$.

3.5. Image Overlay Application

The rendering of the advected image and the noise blending may be followed by an optional image overlay. An overlay enhances the resulting texture-based representation of

surface flow by applying color, shading, or any attribute mapped to color (Figure. 5, bottom, middle). The overlay is constructed once for each static scene and applied after the image advection, edge blending, and noise blending phases. Since the image advection exploits frame-to-frame coherency, the overlay must be applied after the advection in order to prevent the surface itself from being smeared.

4. Texture-Based Flow Visualization on Isosurfaces

Here we describe the way in which we apply the method described in Section 3 to isosurfaces. Specifically, we describe ways to address: (1) the normal component of the flow to the isosurface, (2) perceptual challenges associated with viewing flow on isosurfaces, (3) issues related to resampling the vector field, and (4) some implementation details.

4.1. Applying a Normal Mask

When visualizing flow on normal boundary surfaces like the ring in Figure 5 bottom, right, the direction of the flow generally coincides with the surface itself. As the flow approaches the boundary, it is not allowed to pass through and is pushed in a tangential direction, i.e., it can be described as surface aligned flow. However, in the case of isosurfaces this is no longer true. The flow at an isosurface can sometimes exhibit a strong flow that is normal to the surface, e.g., cross-surface flow. The same also holds true for the case of arbitrary clipping geometries such as those used by Rezk-Salama et al. [11] Simply advecting texture properties according to the vector field projected onto the isosurface could be considered misleading. Is there a way in which this cross-surface component of the flow can be incorporated into the result visualization?

Battke et al [1], who applied LIC to surfaces, address this problem by varying the length of the convolution filter according to the magnitude of the vector component tangential to the surface. In areas where the vector field is oriented almost perpendicular to the surface only very little smearing of the texture occurs, i.e., the input noise is visible instead of a convolved texture. Our approach is required to be consistent with the visualization of flow on boundary surfaces. When we apply texture-based flow visualization to boundary surfaces, the amount of texture-smearing indicates velocity magnitude, i.e., texture is smeared into longer streaks in areas of higher velocity magnitude. We don't want to change the semantic interpretation of smearing for isosurfaces.

We propose an idea inspired by the well known velocity mask, namely, a *normal mask*. A velocity mask can be used to dim or highlight high frequency noise in low velocity regions. Whereas, a normal mask can be used to dim regions of the vector field that have strong cross-flow component to the isosurface. We propose the following to define the normal mask:

$$\alpha = (\bar{\mathbf{v}} \cdot \mathbf{n})^m \tag{4}$$

where α increases as a function of the product of the velocity, $\mathbf{v}$, and normal vector to the surface, $\mathbf{n}$, at that point. Here, m is an arbitrary number. In practice, it is typically around unity giving the opacity a nearly linear behavior. In our case, the image overlay (Figure 5 bottom, middle) becomes more opaque in regions with a strong cross-flow component and more transparent in areas of highly tangential velocity. With the normal mask enabled, the viewer's attention is drawn away from areas of strong cross-flow component, and towards areas of high tangential velocity. Note however, that the texture properties are still advected according the velocity vectors projected onto the isosurface.

The result of applying a normal mask to the ring surface from Figure 5 is shown in Figure 6. The inlet area where the flow is generally orthogonal to the surface has a high opacity covering up the high spatial frequency texture. Note that if we encode the normal mask as opacity, another simulation attribute can be mapped to hue. In our application this is a requirement for consistency.

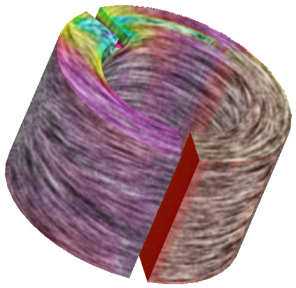

Figure 6: *The result of applying a higher contrast normal mask to the ring from Figure 5. The texture is no longer visible at the inlet of the ring where the texture reflects the flow orthogonal to the surface.*

4.2. Normal Mask Implementation

We can integrate the implementation of the normal mask with very little overhead by taking advantage of the graphics hardware. If we look closely at the construction of the velocity image in Section 3.1 we note that the the *R*, *G*, and *B* image channels are used to encode the *x*, *y*, and *z* values of the vector components at each vertex defining the surface. This leaves the alpha channel as a free parameter in the velocity image construction. In order to implement the normal mask, we simply store the result of $\bar{\mathbf{v}} \cdot \mathbf{n}$ into the alpha channel when rendering the velocity image, at the same time we are storing the *x*, *y*, and *z* vector components. And when reading back the image buffer, reading the alpha component in addition to the *R*, *G*, and *B* component comes at very little overhead. Some results of applying this normal mask to an isosurface are shown in Figure 7. We can see that the flow at the isosurface just below the intake port in the foreground (in white) has a strong normal component to the isosurface. The higher

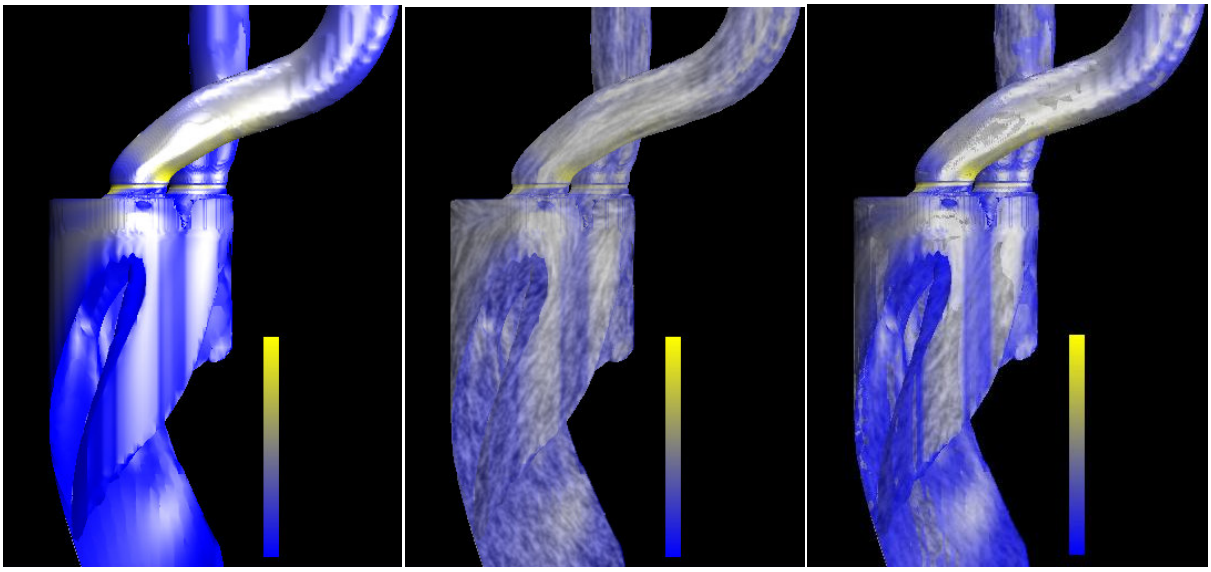

Figure 7: *From left to right: (left) a velocity isosurface of value 5.0 m/s with a CFD simulation attribute mapped to hue, (middle) texture-based flow visualization on the isosurface (right) texture-based flow visualization on the isosurface combined with a normal mask. A close-up is shown in Figure 8.*

frequency texture in this region is difficult to see. Figure 8 shows a close-up of Figure 7 for increased clarity of exposition. Note also that we have chosen a simpler color scale in this case to reduce the visual complexity of the result. We find that using using a full range of hue for the color mapping (like in Figure 2) in combination with variable opacity for the normal mask is visually complex. So we provide the option of trading off some complexity in the color map for including the normal mask.

4.3. Perceptual Issues

Figure 9, top, shows a close-up view of a velocity isosurface with texture-based flow visualization applied. One perceptual problem with the result is that of occlusion. There is more structure to this isosurface than we can see. Perceptual problems such as occlusion and visual complexity are common to generally all 3D visualizations. One way we addressed this is by implementing an interactive clipping plane.

The clipping plane allows the user to see occluded parts of the isosurface by removing sub-sets of the geometry on one side of the plane, in this example, the side closer to the viewer. Again, the users are interested in cutting planes because of their familiarity. Figure 9, bottom, shows another view of the isosurface with a clipping plane being used. New structures in the isosurface are revealed, namely the structure resulting from flow around an intake port valve.

Of course another alternative is for the engineer to take a 2D slice through the volume, rather than creating an isosurface. This is essentially trading off dimensionality in order to reduce perceptual problems. In our application, the user has both options.

Another option for the user in our application is the use of arbitrary clipping geometries. For example, the user can define a clipping geometry in the shape of a sphere or cylinder and apply the texture-based flow visualization. Again however, this is a trade-off. We may gain by lowering visual complexity and occlusion, but we lose some information about the behavior of the flow, namely, that visualized by the isosurface.

4.4. A Resampling Point of View

One reason this texture-advection method is faster than previous texture-based methods on surfaces is because the injection, blending, and advection of noise textures is done in image space. The key to transforming the three-dimensional nature of textures on surfaces to a two-dimensional problem is via the projection of the vector field to image space, as illustrated in Figure 10.

The vector field from the isosurface is projected onto image space via the velocity image described in Section 3.1. Then, the image is warped according to a regular, rectilinear mesh as described in Section 3.2 and illustrated on the right in Figure 10. By distorting the image according the projected velocity vectors located at the grid intersections, we are implicitly resampling the vector field. This resampling implies some consequences. Unlike the nature of boundary surfaces, isosurfaces may contain very small, disconnected pieces. In

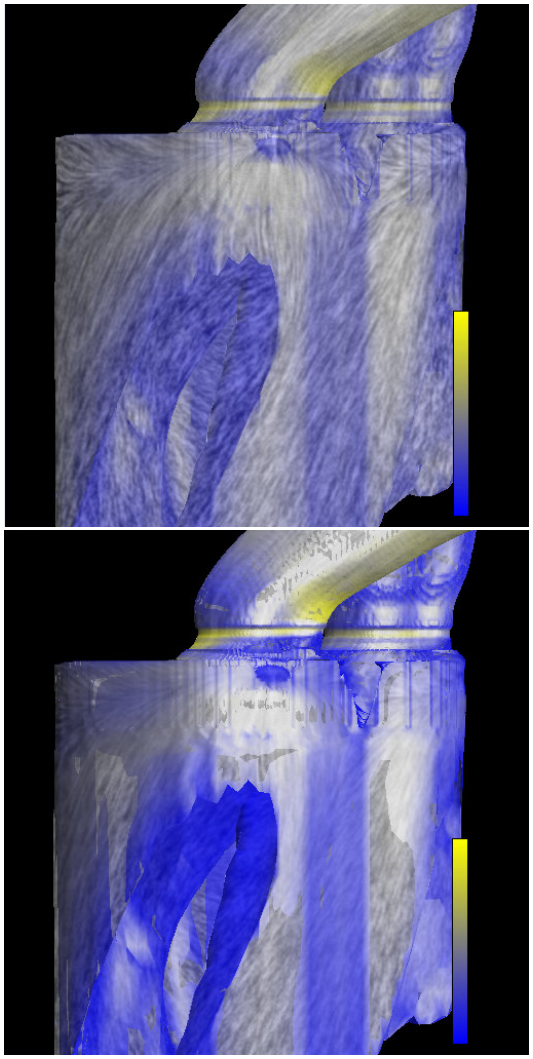

Figure 8: *A close-up of the isosurface from Figure 7: (top) texture-based flow visualization on the isosurface (bottom) texture-based flow visualization on the isosurface combined with a normal mask.*

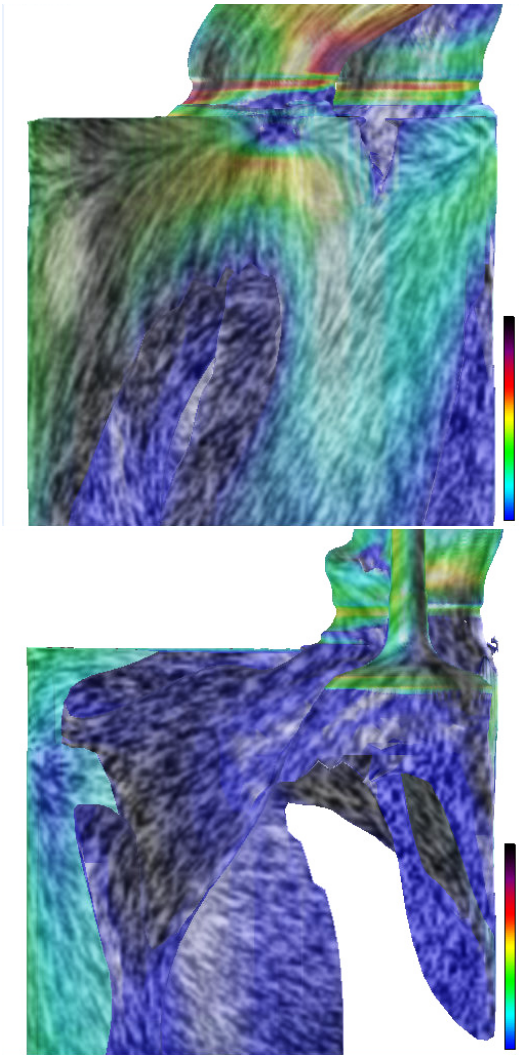

Figure 9: *(top) a close-up of a velocity isosurface of value 5.0 m/s with no clipping plane (bottom) the same isosurface shown using a clipping plane tangent to the view-point in order to reveal occluded isosurface structures*

some cases these pieces may only cover a few pixels. This implies that we need a high vector field resampling rate when advecting the textures in image space. In other words, the sampling to pixel ratio should be small, e.g., sampling at every pixel or every other pixel. In order to handle this, we give the user the option of different advection grid resolutions. In our implementation, the highest grid resolution samples the vector field at every pixel, while the second highest advection grid resolution samples the vector field at every other pixel.

Another reason we give the user control over the resolution of the advection grid is because we want to retain the advantages obtained by decoupling the original surface mesh with the mesh used to advect the textures. This decoupling prevents computation on those polygons whose area covers less than one pixel. And in the case of Figure 3 there are thousands of such polygons. We note also that zooming in on a surface implicitly increases the sampling rate of the vector field because more of the image is spread out while the resolution of the advection grid at the same time remains the same.

The fact that an isosurface may contain many small, disconnected pieces also implies that we need a high frequency texture in the spatial domain. In our implementation, we give

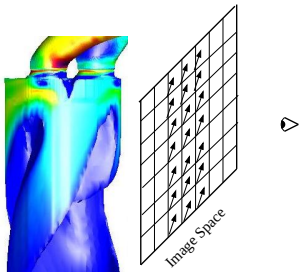

Figure 10: *Image space based flow visualization from a resampling point of view. Setting up a regular, rectilinear advection grid in image space (right) implies a vector field resampling process. The advection mesh shown here is of very coarse resolution for illustration.*

the user control over the spatial frequency of the noise injection. Some samples of these spatial frequencies are illustrated in Figure 11. Using a high spatial frequency allows for the visualization of flow on even very small, disconnected pieces of an isosurface.

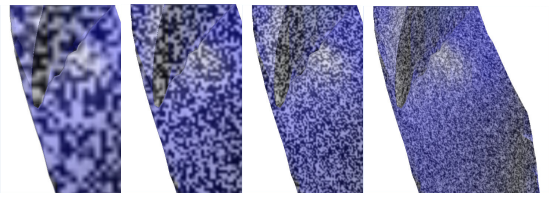

Figure 11: *Different spatial frequency noise that can be used in the noise injection and advection process: Small, disconnected pieces of an isosurface imply that a high spatial frequency noise should be used.*

5. Results and Discussion

If we take a closer look at Figures 8 and 9, we can see that the texture-based flow visualization provides additional insight into the behavior of the flow. One of the questions that the engineer poses is: Where in the volume is the ideal swirl flow pattern *not* being met? Within the texture, we can see that the ideal swirl flow pattern is not being met in just below the intake ports themselves. Namely, we can see that two areas of the flow are working *against* each other just beneath the actual boundary surface of the combustion chamber. This is shown more clearly in a close-up in Figure 12. The normal mask in Figure 12 highlights the boundary between this destructive flow pattern. This is contrasted with only the isosurface itself (Figure 2) where area destructive flow is not

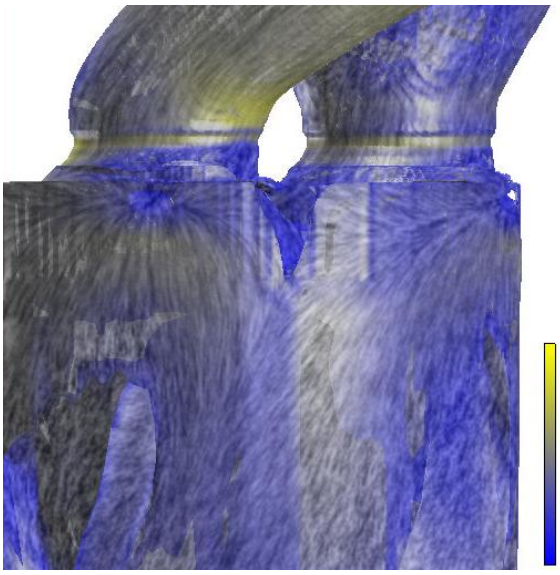

Figure 12: *A close-up of a velocity isosurface of value 5.0 m/s with texture-based flow visualization and a normal mask applied. With the texture advection on the isosurface, it is clear that the ideal swirl flow pattern is not exhibited in this region.*

obvious. The destructive flow pattern is made even more obvious in an animation of the flow. †

Figure 13 shows the isosurface from a top point of view. In this view, it is clear that much of the flow is consistent with the swirl flow pattern. Again, this is even clearer when watching an animated sequence of the visualization result.

Figure 14, top, shows the intersection of three blood vessels. The larger vessel on the right brings in blood and distributes it to two small vessels on the left. An abnormal pocket, e.g., an aneurysm, has developed at the junction of the three vessels. The observer may be interested to investigate the inside of the pocket to see the resulting blood flow pattern. If we look at the blood flow at the surface, as in Figure 14, top, we see mostly noise. The velocity of the blood flow at the surface of the pocket is moving very slow relative to the vessel surfaces.

Figure 14, bottom, shows a velocity isosurface (of 0.04 m/s) inside the volume with texture-based flow visualization applied. Shown clearly is the recirculation zone in the pocket with blood flowing upstream (in the opposing direction). This second example was chosen in an effort to support our claim that the hybrid approach of texture-based flow

† Supplementary MPEG animations and images of the results can be found at:
`http://www.vrvis.at/ar3/pr2/VisSym04/`

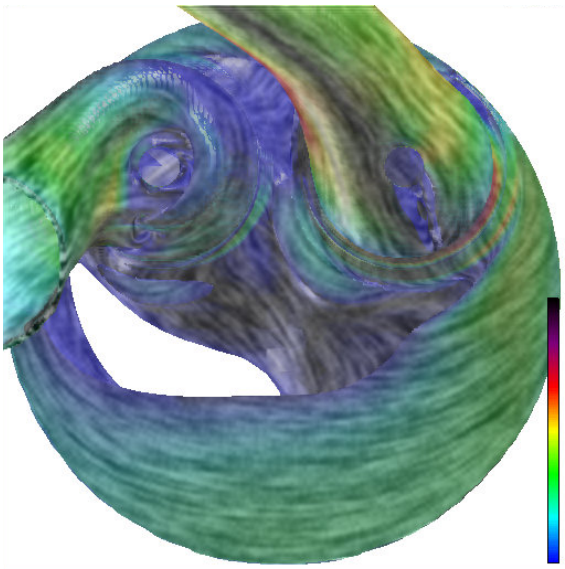

Figure 13: *A view from the top of the velocity isosurface of value 5.0 m/s with texture-based flow visualization applied. From this view, it appears as if most of the flow is consistent with the swirl flow pattern.*

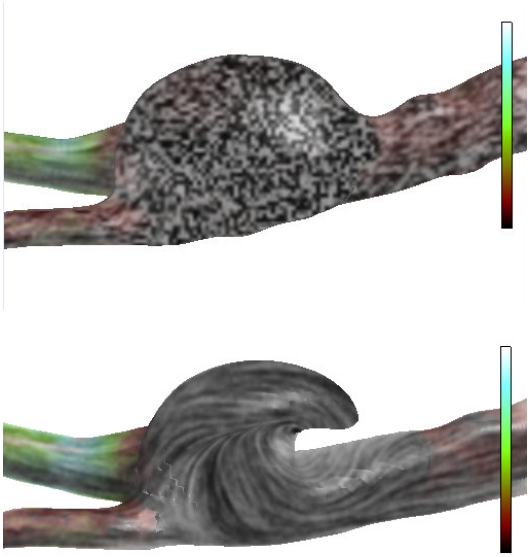

Figure 14: *(top) the intersection of three blood vessels. An abnormal pocket has formed at the junction. (bottom) a velocity isosurface of value 0.04 m/s with texture-based flow visualization applied. The recirculation zone where blood flows in the opposing direction becomes clear.*

visualization on isosurfaces can be useful, not only in the automotive domain.

6. Performance

Performance was evaluated on a PC with an Nvidia 980XGL Quadro graphics card, with a 2.8 GHz dual-processor and 1 GB of RAM. The performance times reported in Table 1 support interactive exploration of flow on isosurfaces. This is important for the case of changing isovalues. When the user changes the isovalue, texture updates only require a fraction of a second. And the transition is generally coherent because each frame is blended with the previous frame as described in Section 3.4.

The first time reported in the FPS column is for the static case, i.e., the absence of changes to the view point. The times shown in parenthesis indicate the dynamic cases of interactive zooming, rotation, and translation of the view point. More specifically, the dynamic cases require the construction of a velocity image, image overlay, as well as geometric edge detection. The reported times are about three times faster than those reported by Laramee et al. [6]. This is mainly due to the updated hardware used for the evaluation and some small improvements to the implementation.

The performance time of the algorithm depends on the resolution of the mesh used to perform the advection and the number of polygons in the original isosurface mesh. In the static case, the algorithm no longer depends on the number of polygons in the surface mesh, but on the area covered in

num. polygons	advection mesh res.	FPS
10K	64^2	64 (35)
	128^2	64 (18)
	256^2	32 (8)
	512^2	15 (2.3)
48K	64^2	64 (13)
	128^2	64 (10)
	256^2	32 (6)
	512^2	15 (2)
221K	64^2	64 (4)
	128^2	64 (4)
	256^2	32 (2.9)
	512^2	13 (1.5)

Table 1: *Sample frame rates for the visualization algorithm.*

image space and the resolution of the advection mesh. In all the performance numbers given in Table 1 approximately 75% of image space is covered. We recommend that the user explore the data with a lower resolution advection mesh and then increase the resolution when higher accuracy analysis or presentation is required.

Normal mask construction does not introduce significant

overhead since it is easily built into the advection process itself. For example, the isosurface shown in Figures 12 and 13 is composed of 243K polygons. In the static case, the normal mask has no effect on frame rates. They are the same as those listed in Table 1. In the dynamic case using a 128^2 advection mesh, the frame rate drops from 3.3 to 3.0 FPS with the addition of the normal mask.

7. Conclusions and Future Work

We have shown how the image space based texture-advection technique of Laramee et al. [6] can be applied to isosurfaces. Isosurfaces provide information into the 3D characteristics of flow that 2D slices and boundary surfaces alone cannot. We have shown that adding the texture-based representation of flow to isosurfaces can give the engineer new insight into the behaviour of swirl flow when examining CFD simulation data. We have also applied the method to the visualization of blood flow. Furthermore, the method is fast and supports user interaction such as zooming, rotation, and translation.

Future work can go in many directions including visualization of texture-based flow visualization on time-dependent isosurfaces, streamsurfaces, and unsteady 3D flow. Challenges will include both interactive performance time and perceptual issues. Future work also includes the application of more specialized programmable graphics hardware features in the manner of Weiskopf et al. [16, 17]

The author(s) thank all those who have contributed to this research including AVL (`www.avl.com`) and the Austrian governmental research program called Kplus (`www.kplus.at`). We extend a special thanks to Markus Hadwiger of the VRVis Center For Virtual Reality and Visualization (`www.VRVis.at`) for his technical support. We also thank Jarke van Wijk for his encouraging dialog and Helmut Doleisch for helping to prepare the final manuscript. All CFD simulation data shown in this paper has been provided courtesy of AVL.

References

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

[2] B. Cabral and L. C. Leedom. Imaging Vector Fields Using Line Integral Convolution. In *Poceedings of ACM SIGGRAPH 1993*, Annual Conference Series, pages 263–272. ACM Press / ACM SIGGRAPH, 1993.

[3] W.C. de Leeuw and J.J. van Wijk. Enhanced Spot Noise for Vector Field Visualization. In *Proceedings IEEE Visualization '95*, pages 233–239. IEEE Computer Society, October 1995.

[4] L. K. Forssell and S. D. Cohen. Using Line Integral Convolution for Flow Visualization: Curvilinear Grids, Variable-Speed Animation, and Unsteady Flows. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):133–141, June 1995.

[5] R. S. Laramee and R. D. Bergeron. An Isosurface Continuity Algorithm for Super Adaptive Resolution Data. In John Vince and Rae Earnshaw, editors, *Advances in Modelling, Animation, and Rendering: Computer Graphics International (CGI 2002)*, pages 215–237, Bradford, UK, July 1-5 2002. Computer Graphics Society, Springer.

[6] R. S. Laramee, B. Jobard., and H. Hauser. Image Space Based Visualization of Unsteady Flow on Surfaces. In *Proceedings IEEE Visualization '03*, pages 131–138. IEEE Computer Society, 2003.

[7] W. E. Lorensen and H. E. Cline. Marching Cubes: a High Resolution 3D Surface Construction Algorithm. In *SIGGRAPH '87 Conference Proceedings (Anaheim, CA, July 27–31, 1987)*, pages 163–170. Computer Graphics, Volume 21, Number 4, July 1987.

[8] X. Mao, M. Kikukawa, N. Fujita, and A. Imamiya. Line Integral Convolution for 3D Surfaces. In *Visualization in Scientific Computing '97. Proceedings of the Eurographics Workshop*, pages 57–70. Eurographics, 1997.

[9] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. Feature Extraction and Visualization of Flow Fields. In *Eurographics 2002 State-of-the-Art Reports*, pages 69–100, Saarbrücken Germany, 2–6 September 2002. The Eurographics Association.

[10] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The State of the Art in Flow Visualization: Feature Extraction and Tracking. *Computer Graphics Forum*, 22(4), 2003. forthcoming.

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

[12] D. Stalling. LIC on Surfaces. In *Texture Synthesis with Line Integral Convolution*, pages 51–64. ACM SIGGRAPH 97, International Conference on Computer Graphics and Interactive Techniques, 1997.

[13] G. M. Treece, R. W. Prager, and A. H. Gee. Regularised Marching Tetrahedra: Improved Isosurface Extraction. *Computers and Graphics*, 23(4):583–598, August 1999.

[14] J. J. van Wijk. Spot noise-Texture Synthesis for Data Visualization. In Thomas W. Sederberg, editor, *Computer Graphics (Proceedings of ACM SIGGRAPH 91)*, volume 25, pages 309–318. ACM, 1991.

[15] J. J. van Wijk. Image Based Flow Visualization. *ACM Transactions on Graphics*, 21(3):745–754, 2002.

[16] D. Weiskopf, G. Erlebacher, M. Hopf, and T. Ertl. Hardware-Accelerated Lagrangian-Eulerian Texture Advection for 2D Flow Visualizations. In *Proceedings of the Vision Modeling and Visualization Conference 2002 (VMV-01)*, pages 439–446, November 21–23 2002.

[17] D. Weiskopf, M. Hopf, and T. Ertl. Hardware-Accelerated Visualization of Time-Varying 2D and 3D Vector Fields by Texture Advection via Programmable Per-Pixel Operations. In *Proceedings of the Vision Modeling and Visualization Conference 2001 (VMV 01)*, pages 439–446, November 21–23 2001.