3 Animation of Orthogonal Texture Patterns

The method from the previous chapter describes how vector fields can be visualized that are technically challenging in terms of size or complexity. However, another aspect requires attention as well: how well can a human observer perceive animations in visualization? In general, animation has been used successfully for vector field visualization because it can show the direction, orientation, and especially the magnitude of a vector field. In addition, animation can mitigate curve intersection issues that occur for long path lines or streak lines of a time-dependent flow. However, there is a lack of related work that would have considered the perception of such animations for flow visualization. A recent overview of the role of perception in visualization and computer graphics is presented by Healey and Enns [HE12], however, the perception of animated patterns is not discussed in their work.

Based on results from vision research, it can be stated that previous approaches like animated stream lines are non-optimal for local motion perception. Texture-based methods significantly reduce spatial frequency along integral curves to display those curves. However, there is substantial evidence for a spatial frequency tuning of the motion detectors in our human visual system (HVS), and optimal spatial frequencies are typically much higher than the spatial frequencies produced by texture-based methods (see Section 3.2.2).

Therefore, the approach presented in our publication [BW08a] is to decouple the direction of the line-like patterns from the direction of animation. More specifically, an orthogonal vector field should be used (i.e., the original vector field rotated by $\pi/2$) to construct line-like patterns and the original vector field is used to drive the animation. In this way, the spatial frequencies along the direction of motion are determined by the spatial frequencies of the input noise (for LIC or texture advection), which are independent of the length scales along the line patterns (controlled by the filter length of LIC or texture advection). This visualization approach resembles a moving wave front of the vector field and therefore provides an intuitive analogy to the real world.

This method of the orthogonal vector field and its corresponding visualization is denoted as *ortho-vis*. In contrast, the traditional texture-based visualization of stream lines or path lines is called *tangent-vis* to indicate that it shows the tangential direction of the vector field. Ortho-vis and tangent-vis can be combined to a single visualization image that retains the perceptual benefits of ortho-vis, while providing the traditional, familiar visualization via tangent-vis.

3.1 Algorithm Overview

This method targets vector field visualization by combining texture-based representations of the tangential and orthogonal vector field directions. The algorithm computes intermediate results of ortho-vis and tangent-vis elements in an essentially independent way, combining those results in a final image-compositing step. Figure 3.1 provides a high-level view on the data flow and computational components of the complete algorithm. Temporally coherent animation starts with generating coherently moving noise images in the first part of the algorithm. Then, the process is split in the independent computation of ortho-vis and tangent-vis. Finally, image compositing results in a combined visualization of ortho-vis and tangent-vis. On this abstract level, the above algorithm works for 2D data sets (i.e., on planar 2D manifolds) and 2.5D data sets (i.e., on curved surfaces embedded in 3D space) alike. Since an image-space approach is used for 2.5D visualization, the operations of this approach are sufficient to handle 2.5D vector fields—these operations are designed to work in image-space as well. In Figure 3.1, these operations are located in the components labeled “orthogonal vis”, “tangential vis”, and “compositing”. Although the first component, “noise transport” is designed to work in object-space, it can be implemented for both 2D and 2.5D data sets.

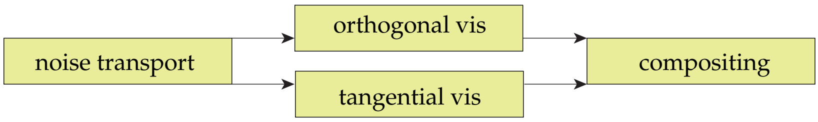

Figure 3.1: High-level illustration of the structure of the complete visualization process.

3.2 Orthogonal Vector Field Representation

The orthogonal vector field representation (ortho-vis) is introduced first in a formal, mathematical way and then the new visualization method is motivated by showing similarities to existing visualization approaches and by providing a perceptual rationale. The subsequent parts of this section discuss the animation and temporal coherence of moving visualization patterns.

3.2.1 Spatial Patterns

As introduced in Section 1.3, a tangential vector field $\vec{v}$ can be defined on a smooth and orientable 2D manifold M. Since M is orientable, an operator Ω can be defined that rotates a vector within the tangent plane by $\pi/2$. The inverse operator Ω^{-1} yields a rotation by $-\pi/2$. The orthogonal vector field $\vec{u}$ is defined as

$$\vec{u} : M \longrightarrow TM\,, \quad \vec{x} \longmapsto \Omega\vec{v}(\vec{x}).$$

The idea is to display the orthogonal vector field $\vec{u}$ instead of the original vector field $\vec{v}$. The actual visualization relies on integral curves (stream lines in the context of flow visualization, field lines in the context of electric, magnetic, or related fields) to show the direction of $\vec{u}$. In this thesis, the focus lies on the texture-based visualization of stream lines by means of LIC (see Sections 3.3 and 3.4), but other methods such as geometrically constructed stream lines might also be employed. Figure 3.2 illustrates an example of $\vec{u}$ and $\vec{v}$ by means of a few geometric lines that represent integral curves. So far, a stationary vector field has been assumed. For a time-dependent vector field, the above approach is applied to an instantaneous vector field for a given time in order to produce a single visualization for that time.

Before animation can be considered—the main aspect of this visualization approach—the use of the rotated vector field for a single frame of the visu-

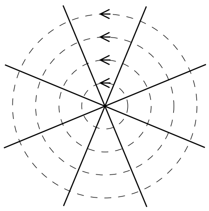

Figure 3.2: Illustration of two perpendicular families of lines: for a circular vector field $\vec{v}$ (dashed lines) and a radial vector field $\vec{u}$ (solid lines).

alization will be motivated. First, the mapping by the rotation operator Ω is one-to-one, i.e., the original vector field $\vec{v}$ can be recovered by applying the uniquely defined inverse operator Ω^{-1}. While this argument shows that the same information content is displayed by $\vec{u}$ and $\vec{v}$, the question remains how effective the rotated vector field is for visualization purposes. Here, the special choice of the $\pi/2$ rotation angle becomes important because analogous uses of perpendicular line structures are well known and accepted in visualization. One analogy comes from the representation of 2D scalar fields by either contour lines (isolines) or gradient directions: gradients and contours are perpendicular by construction. For example, Figure 3.2, which was used to illustrate an orthogonal vector field, can also be interpreted as a visualization of a scalar field with maximum value in its center, concentric circles as contour lines (dashed), and radial gradient lines (solid). A related analogy is based on the Helmholtz decomposition of vector fields [Her58]. Adopting the notation of Polthier and Preuß [PP00], a vector field $\vec{v}$ on a 2D manifold can be written as

$$\vec{v} = \nabla\phi + \Omega\nabla\omega + \vec{\eta},$$

with the curl-free part $\nabla\phi$, the divergence-free part $\Omega\nabla\omega$, and the remaining harmonic part $\vec{\eta}$. The scalar functions ϕ and ω serve as potentials for the gradient field $\nabla\phi$ and the co-gradient field $\Omega\nabla\omega$ (defined as the gradient rotated by $\pi/2$).

Publications on the Hodge-Helmholtz decomposition have focused on variational, discretized computations for triangulated grids [PP00, PP03, TLHD03]. Assuming a divergence-free vector field represented by ω, the orthogonal vector field approach shows the field lines of the gradient $\nabla\omega$. Similarly, a curl-free vector field based on ϕ would show the gradient $\nabla\phi$ by traditional visualization methods. Therefore, the orthogonal vector field visualization could be regarded as the “dual” of the traditional flow visualization. Another analogy can be found in the propagation of wave fronts. The Eikonal equation [Gol80], which, for example, applies to the propagation of action in the Hamilton-Jacobi model of classical mechanics or to light propagation, describes this propagation in terms of fronts that are orthogonal to the propagation direction, i.e., those fronts are analogues of ortho-vis, whereas the propagation vectors are analogues of tangent-vis. A related physical analogy is the propagation of water waves, where the wave fronts are perpendicular to wave propagation.

3.2.2 Animation and Motion Perception

So far, only static visualization by a single image was discussed. Although the spatial patterns in one frame are important, animation plays an even more crucial role in the proposed approach. The basic idea is to drive the animation by the original vector field $\vec{v}$, i.e., the direction of motion (given by $\vec{v}$) and the integral curves in an image (determined by $\vec{u}$) are perpendicular. Figure 3.2 illustrates this approach: the solid, radial lines show the curves of $\vec{u}$, which are transported along the circular flow $\vec{v}$, leading to a counter-clockwise rotation.

Why is the decoupling of temporal evolution and spatial patterns useful? The main motivation comes from research on human visual perception. There is indication for a spatial frequency tuning of the HVS: how well we perceive motion depends on the spatial frequency of moving patterns. Low-level motion perception is based on small receptive fields that serve as local motion detectors (see, e.g., [AB87]). Vision research and physiological investigations have addressed various aspects of motion perception, including the detection and discrimination of moving patterns, the influence of contrast and color, and the breakdown of the perception of coherent motion under certain conditions.

Although this topic is still an area of active research, a general observation of a frequency tuning of motion detection can be found in the literature. One interesting aspect of recent studies is that the characteristics of receptive fields may be adaptive—dependent on the stimulus. For example, Cavanaugh et al. [CBM02] describe that at low contrast, a wider spatial region (with less surround suppression) is used as input to increase sensitivity, whereas a high-contrast stimulus leads to higher spatial resolution using increased surround suppression. This principal observation can also be found in the context of motion perception [TL05], where high contrast favors the detection of high-frequency stimuli and low contrast favors large stimuli. Tadin and Lappin [TL05] report an optimal size of 0.5 deg (degrees with respect to the subtended angle as seen by the viewer) for a high contrast of 92%. Typically, texture-based vector field visualization uses patterns of high luminance contrast and, therefore, high spatial frequency patterns are appropriate. Another observation is that local and global motion detectors can be distinguished (see, e.g., [BD02]).

In this section, the focus lies on local motion detection, which is optimal for certain spatial frequencies. Bex and Dakin [BD02] report a maximum sensitivity for local motion detection for spatial frequencies around 2 cycles/deg. A similar number of 3 cycles/deg is given by Watson and Turano [WT95] as optimal motion stimulus. The actual value for the optimal spatial frequency of patterns depends on several outside parameters, but many studies agree upon frequencies somewhere around or above 2 cycles/deg. In contrast to these recommendations, typical visualization applications show extended stream lines of 100–200 pixels length. In other words, depending on viewing distance and screen resolution, commonly used spatial patterns deviate from the optimum by a factor of 5–10.

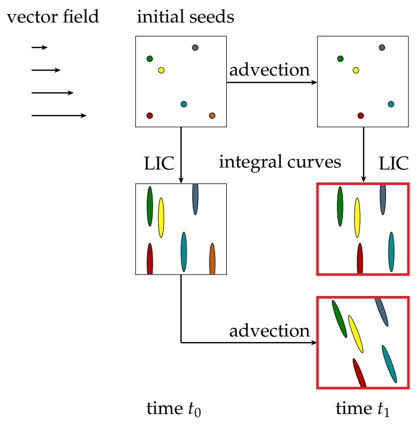

Figure 3.3: Illustration of two transport and visualization approaches, applied to a shear flow. The two resulting images (outlined in red) differ because LIC and advection are not commutative. Seeds and stream lines are colored to allow for an easier recognition of correspondence between images.

3.2.3 Temporal Coherence

The intent of the ortho-vis method is to transport integral curves of the rotated vector field $\vec{u}$ along the original vector field $\vec{v}$ in order to control the spatial frequency of the transported patterns along the transport direction. One has to keep in mind that the vector fields may be time-dependent. This transport could be realized by first constructing integral curves of $\vec{u}$ for an initial time t_0 and then advecting those curves along $\vec{v}$ to a later time t_1. An alternative way is to first advect the seed points (i.e., initial noise for LIC) along $\vec{v}$ from time t_0 to t_1 and then construct the integral curves of $\vec{u}$ for time t_1. Figure 3.3 illustrates both approaches for the example of a shear flow. Unfortunately, these two transport approaches do not necessarily lead to the same result, as demonstrated in Figure 3.3. The first approach guarantees temporal coherence of the transported integral curves because the curves themselves are advected. The second approach makes sure that the integral curves are always perpendicular to $\vec{v}$. Since the two approaches may lead to different results, it is impossible to have a mechanism that maintains orthogonal vector field lines and achieves temporal coherence at the same time.

In the following, mathematical expressions are derived that quantitatively describe the inconsistency between the advection of orthogonal LIC lines and the from-scratch construction of those lines. Those expressions are used to provide the motivation of temporal filtering that is applied in the ortho-vis approach

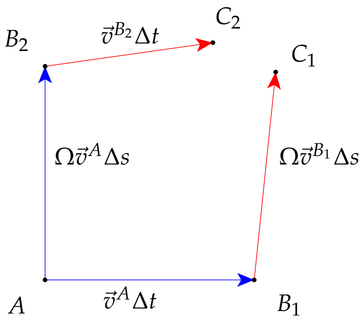

Figure 3.4: Points and transport vectors used for the derivation of inconsistent transport of orthogonal lines.

to overcome the issues of temporal incoherence.

To begin, the distance is derived between a particle transported by the original steady vector field $\vec{v}$ and a particle transported by the orthogonal vector field $\vec{u} = \Omega\vec{v}$. Later, this result is applied to the specific case of advected orthogonal LIC lines. The derivation targets a description by means of differentials; therefore, the derivation utilizes first-order Taylor expansions to derive mathematical expressions, which is completely consistent with final results based on differential quantities.

The following notation is used for the mathematical expressions: uppercase letters label points in 2D space, and those labels are used as superscripts attached to respective variables. One variable is the position of a point, denoted by $\vec{p}$; i.e., $\vec{p}^{A}$ is the position of point A. 2D vector field values are denoted $\vec{v}$; i.e., $\vec{v}^{A}$ is the vector field at point A. Components of points and vectors are indicated by subscripts $_x$ and $_y$, respectively. To denote the approximation that is introduced by using Taylor expansion, the approximation sign "$\approx$" is used. As illustrated in Figure 3.4, the following points related by first-order Euler integration for particle transport are considered:

$$\vec{p}^{A}$$

$$\vec{p}^{B_1} \approx \vec{p}^{A} + \vec{v}^{A}\Delta t \tag{3.1}$$

$$\vec{p}^{B_2} \approx \vec{p}^{A} + \Omega\vec{v}^{A}\Delta s \tag{3.2}$$

$$\vec{p}^{C_1} \approx \vec{p}^{B_1} + \Omega\vec{v}^{B_1}\Delta s \tag{3.3}$$

$$\vec{p}^{C_2} \approx \vec{p}^{B_2} + \vec{v}^{B_2}\Delta t. \tag{3.4}$$

Point A is the starting point for the following discussion. The step sizes Δt and

Δs correspond to transport along the vector field $\vec{v}$ and the rotated vector field $\Omega\vec{v}$, respectively. The rotation operator is defined in component-wise notation as

$$\Omega\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} -v_y \\ v_x \end{pmatrix}.$$

First-order Taylor expansion of the vector field around the point A leads to

$$\vec{v}^{B_1} \approx \vec{v}^A + (J\vec{v}^A)\,(\vec{p}^{B_1} - \vec{p}^A) \approx \vec{v}^A + (J\vec{v}^A)\,(\vec{v}^A\Delta t) \tag{3.5}$$

and

$$\vec{v}^{B_2} \approx \vec{v}^A + (J\vec{v}^A)\,(\vec{p}^{B_2} - \vec{p}^A) \approx \vec{v}^A + (J\vec{v}^A)\,(\Omega\vec{v}^A\Delta s), \tag{3.6}$$

with the Jacobi matrix

$$J\vec{v}^A = \begin{pmatrix} \frac{\partial v_x^A}{\partial x} & \frac{\partial v_x^A}{\partial y} \\ \frac{\partial v_y^A}{\partial x} & \frac{\partial v_y^A}{\partial y} \end{pmatrix}.$$

Without loss of generality, a Frenet frame [Gal01] is used along the stream line of $\vec{v}$ as coordinate system. One axis of the Frenet frame points along the tangent direction $\vec{v}$, the other axis points along the perpendicular direction $\Omega\vec{v}$. Any orthogonal coordinate system can be transformed to the Frenet frame by rotation. After this rotation, the x axis is aligned with $\vec{v}$ and the y axis is aligned with $\Omega\vec{v}$. In this coordinate system,

$$\vec{v}^A = \begin{pmatrix} v_x^A \\ 0 \end{pmatrix}.$$

Using this coordinate system, Equations 3.5 and 3.6 lead to the component-wise expressions

$$\Omega\vec{v}^{B_1} \approx \begin{pmatrix} 0 \\ v_x^A \end{pmatrix} + \begin{pmatrix} -\frac{\partial v_y^A}{\partial x} \\ \frac{\partial v_x^A}{\partial x} \end{pmatrix} v_x^A \Delta t \tag{3.7}$$

and

$$\vec{v}^{B_2} \approx \begin{pmatrix} v_x^A \\ 0 \end{pmatrix} + \begin{pmatrix} \frac{\partial v_x^A}{\partial y} \\ \frac{\partial v_y^A}{\partial y} \end{pmatrix} v_x^A \Delta s. \tag{3.8}$$

Using the Taylor expansions for the vector field from Equations 3.7 and 3.8, the position expressions from Equations 3.1–3.4 can be combined to yield the difference vector

$$\begin{aligned}
\vec{p}^{C_2} - \vec{p}^{C_1} &\approx \vec{p}^A + \begin{pmatrix} 0 \\ v_x^A \end{pmatrix} \Delta s + \left[\begin{pmatrix} v_x^A \\ 0 \end{pmatrix} + \begin{pmatrix} \frac{\partial v_x^A}{\partial y} \\ \frac{\partial v_y^A}{\partial y} \end{pmatrix} v_x^A \Delta s \right] \Delta t \\
&\quad - \left\{ \vec{p}^A + \begin{pmatrix} v_x^A \\ 0 \end{pmatrix} \Delta t + \left[\begin{pmatrix} 0 \\ v_x^A \end{pmatrix} + \begin{pmatrix} -\frac{\partial v_y^A}{\partial x} \\ \frac{\partial v_x^A}{\partial x} \end{pmatrix} v_x^A \Delta t \right] \Delta s \right\} \\
&= \begin{pmatrix} \frac{\partial v_x^A}{\partial y} + \frac{\partial v_y^A}{\partial x} \\ \frac{\partial v_y^A}{\partial y} - \frac{\partial v_x^A}{\partial x} \end{pmatrix} v_x^A \Delta s \Delta t \; . \qquad (3.9)
\end{aligned}$$

If the goal would be to construct orthogonal vector field lines that exhibit velocity-dependent markers along those lines, then Equation 3.9 would appropriately describe the inconsistency between advection of orthogonal lines from the previous time step and from-scratch construction of orthogonal lines for the current time step, i.e., Equation 3.9 describes the difference in transport of point particles along $\vec{v}$ and $\Omega\vec{v}$, respectively. The scaling factors Δs and Δt represent the step sizes for the discretization of transport, and $\vec{v}_x^A$ can be considered as an overall scaling factor for flow velocity. Therefore,

$$\vec{\delta}_{\text{full}} = \begin{pmatrix} \frac{\partial v_x^A}{\partial y} + \frac{\partial v_y^A}{\partial x} \\ \frac{\partial v_y^A}{\partial y} - \frac{\partial v_x^A}{\partial x} \end{pmatrix} \qquad (3.10)$$

is the interesting vector-valued measure for the inconsistencies of particle transport. In fact, the orthogonal vector field is visualized by LIC lines based on the normalized vector field. Put differently, it is not of interest how far $\vec{\delta}_{\text{full}}$ differs in the y direction because a difference in distance along the y direction is compensated by the normalization during the LIC computation. Therefore, only the x component of $\vec{\delta}_{\text{full}}$ is relevant for ortho-vis, leading to a single-component measure for inconsistency in orthogonal LIC transport:

$$\vec{\delta}_{\text{LIC}} = \frac{\partial v_x^A}{\partial y} + \frac{\partial v_y^A}{\partial x} . \qquad (3.11)$$

The term $\frac{\partial v_y^A}{\partial x}$ describes the curvature of the vector field, whereas the term $\frac{\partial v_x^A}{\partial y}$ describes the shear. De Leeuw and van Wijk [dLvW93] present a detailed description and interpretation of those differential quantities of a vector field. Their discussion targets 3D flow, which can be immediately applied to the scenario of 2D flow discussed here by restriction to two coordinates x and y. Please note that a frame of reference is used in which the x axis points along the vector field and the y axis along the perpendicular vector field direction. Therefore, the measure of inconsistency is directly related to the curvature and shear of the vector field. In the example of pure shear, as illustrated in Figure 3.3, δ_{LIC} is given by the non-vanishing shear value only. In contrast, the example of the circular flow (Figure 3.2) does not exhibit any temporal incoherence because effects of curvature and shear cancel exactly. For unsteady flow, additional incoherence effects might be introduced.

As mentioned in the beginning of this section, a two-part temporal filtering process is introduced to overcome the problem of temporal incoherence. The first part is a combination of advection and integral-curve construction: initial seed points are advected along $\vec{v}$ from the initial time t_0 to an intermediate time t_i ($t_0 \leq t_i \leq t_1$); then integral curves of $\vec{u}$ are constructed at time t_i; finally, those integral curves are advected along $\vec{v}$ from t_i to t_1. This overall operation is denoted as T_{t_i}. For a texture-based representation, T_{t_i} takes an initial noise image N as input and yields an image of transported integral curves. The second part applies a temporal filtering process in order to balance the conflicting goals of temporal coherence and orthogonal vector field lines. The actual visualization image at time t_1 is

$$I = \int_{t_0}^{t_1} k(t_i) T_{t_i}(N)\, \mathrm{d}t_i, \tag{3.12}$$

with a filter kernel $k(t)$ normalized according to $\int k(t)\, \mathrm{d}t = 1$. This filtering process trades in clearly defined line-like patterns for a consistent and temporally coherent animation: the width of the filter interval $[t_0, t_1]$ determines the amount of “smearing out” and can be gradually adjusted. In fact, there are many important flow fields that exhibit no or only little inconsistencies and thus would not need any filtering. The circular flow of Figure 3.2 is an example of a completely consistent advection and integral-curve construction.

Animated visualization produces images for increasing end times t_1. For a constant filter width $(t_1 - t_0)$, the start time progresses accordingly. To achieve temporal coherence of the final images, the noise images N need to be temporally coherent for different times t_0, which can be ensured by advecting initial noise images along the vector field $\vec{v}$.

An alternative approach avoids the two-part process: by reducing the length of the orthogonal LIC lines in regions of strong inconsistencies, these inconsistencies are reduced. However, this approach has the disadvantage that static images do not show the “wave fronts” of the vector field to the same degree as in the filtered version.

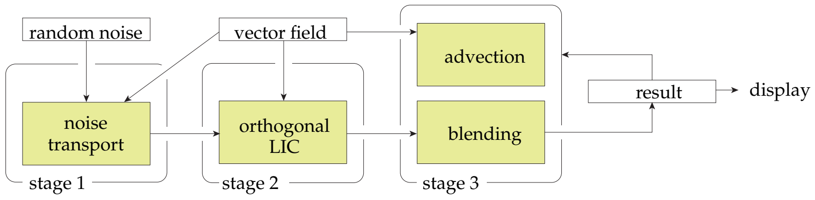

Figure 3.5: Processing stages and data flow for the 2D algorithm.

3.3 2D Algorithm

Here, an algorithm is described that realizes the approach from the previous section for vector fields given on planar 2D domains. All relevant information is 2D (vector field, input noise images, intermediate and final visualization images) and can be represented as 2D images, 2D textures, or 2D uniform grids. Vector data on unstructured, triangulated grids would also work because a triangle mesh can be easily rendered (i.e., rasterized) into a 2D image. The algorithm that produces the final output image can be seen as a pipeline consisting of three major stages (Figure 3.5). Each stage creates an intermediate result that is used as input for the next stage. The first stage (noise transport) implements two aspects of the abstract approach from Section 3.2.3: (1) temporally coherent input noise for different starting times t_0 and (2) the advection of noise from t_0 to the intermediate time t_i. The second stage (orthogonal LIC) constructs a LIC image of the rotated vector field $\vec{u}$ at time t_i. The third stage (advection and blending) implements the transport of LIC patterns from time t_i to t_1 and computes the filter operation from Equation 3.12. In the following, the three stages are explained in more detail.

The first stage is responsible for creating a temporally coherent noise that should move according to the possibly time-dependent vector field $\vec{v}$. Similarly to the approach of Weiskopf et al. [WEE03, Section 4], path lines are traversed from the current time step backward in time in order to accumulate noise injection input from previous times in a Lagrangian manner. This accumulation yields a convolution in time along path lines. The time span of backward particle tracing determines the scale of temporal correlation: typically, some 15–50 integration steps are appropriate for an adequate compromise between computation time and quality of temporal coherence.

The different noise injection images that serve as input for the temporal convolution need to be uncorrelated. To save memory for a large number of noise injection images, they are constructed on-the-fly by reusing a single template image. To achieve this, the template noise image must be periodic (i.e., a seamless texture), which, for example, is automatically achieved by generating a low-pass filtered noise via filtering in Fourier space, using a fast Fourier transform. A new, uncorrelated noise image is produced from the template image by Cranley-Patterson rotation [CP76], which adds the same random shift to each point of the template image. The random shifts are applied on-the-fly while the noise injection texture is accessed. The density of the visual representation is controlled by the characteristics of noise injection. A dense representation is achieved by low-pass filtered white noise; a sparser representation is achieved by Gaussian-filtered sparse input constructed from randomly positioned dots.

Inflow and outflow at boundaries of the domain often cause problems for texture-based methods. This issue is addressed in two ways. First, the injection noise is periodic and thus virtually infinite in size. Second, the vector field is clamped at the boundary, making it virtually infinite as well. Therefore, particles can be traced beyond domain boundaries. Another issue is divergence or convergence of the flow, which could change the spatial frequency of injected noise by stretching or compression. Due to the limited integration length in time (some 15–50 integration steps), this problem does not lead to serious artifacts except for extremely large absolute values of divergence. Finally, the temporal convolution of uncorrelated noise images leads to reduced contrast. The convolution corresponds to a summation of (approximately) independent random variables, resulting in a normal distribution of values according to the central limit theorem. Contrast is restored by histogram equalization.

The result of the first stage is a noise texture that moves along the vector field $\vec{v}$ and serves as input to the second stage. The second stage creates LIC lines that visualize the orthogonal vector field $\vec{u}$ at a fixed time that corresponds to the current visualization time, which is similar to the spatial filtering process described by Weiskopf et al. [WEE03]. The rotation of the original vector field $\vec{v} = (v_x, v_y)$ is computed by a mapping to $\vec{u} = (-v_y, v_x)$. Usual particle tracing and LIC integration are performed with the orthogonal vector field. Vectors are normalized to unit length to obtain LIC lines of equal length. Boundaries of the domain are taken into account by stopping the LIC integration once a particle trace crosses a boundary. Similarly to stage one, contrast is enhanced by histogram equalization.

The third stage of the pipeline transports LIC patterns from the second stage and evaluates the filter operation from Equation 3.12. The goal of the third stage is to produce a temporally coherent and consistent visualization with line patterns that are (approximately) perpendicular to the vector field. Several intermediate images T_{t_i} must be computed and stored to implement the filtering from Equation 3.12. This flexible and accurate approach was constructed mainly for evaluation and comparison purposes. The additional work and memory consumption for generic filtering can be avoided by restriction to an exponential filter kernel, which can be discretized in the form of a recurring application of the *over* operator (i.e., alpha blending with weights α and

$(1-\alpha)$) [EJW05]. The alpha value determines the falloff of the exponential filter. One image used for blending is the result of the second stage; the other image is the visualization result of the previous time step, transported to the current time step by semi-Lagrangian advection. This incremental computation of exponential filtering is mainly recommended for real-time rendering for interactive visualization.

If memory consumption and additional computation time are not limiting factors, then the generic filtering process can be used in order to obtain images of higher quality. The inexpensive, incremental exponential filtering process is compared with generic filter kernels in a parameter study in Section 3.7.3.

A different approach avoids the third stage completely. This approach adapts the length of orthogonal LIC lines in those regions where inconsistencies are present. To make this possible, the inconsistency is precomputed and used as a scaling factor at every point of the vector field to shorten the LIC lines proportional to the amount of inconsistency. The results of this approach are shown in Section 3.7.3.

3.4 2.5D Algorithm

This section describes the visualization of tangential vector fields on curved surfaces embedded in 3D space. The same basic pipeline is adopted as for 2D vector fields (see Figure 3.5), but some modifications and extensions are necessary that are specific to 2.5D data. The following discussion is restricted to those modifications. The input vector field may either be given as a 3D texture that is intersected by the surface or it is attached to the vertices of the surface. Since this algorithm is designed for tangential vector fields, a possibly non-tangential vector field is made tangential by subtracting the normal component of a vector.

The first stage of the pipeline (noise transport) adopts the hybrid physical-/device-space approach of Weiskopf and Ertl which is detailed in Section 1.4.5. This LIC-based technique is turned into temporal convolution by the following modifications. First, particle paths are traced along path lines backward in time only. Here, the vector field is not normalized to unit length. Second, a single input noise is replaced by uncorrelated noise inputs for different times, according to a Cranley-Patterson rotation. Noise is modeled as 3D solid texture in order to achieve temporal coherence even under camera rotations, i.e., noise is attached to the surface geometry in object-space. The result of the first stage is a temporally coherent noise image that moves along the surface of the scene geometry.

The following steps are performed in image-space only, reminiscent of image-space advection techniques [LJH03, vW03]. The second stage (orthogonal LIC) takes the original vector field given in 3D space, rotates it by $\pi/2$ around the local normal vector of the surface, and projects the orthogonal vector field onto image-space. The rotation is determined in object-space by computing the cross product of the surface normal and the tangential vector. The subsequent projection to image-space yields a 2D vector field with respect to image-space coordinates. Finally, LIC is performed in image-space, based on the noise image from stage one and the image-space vector field. Particle tracing for LIC is stopped at the boundaries of the object; the silhouette of the object is determined according to a mask that contains the classification of pixels as foreground or background pixels. Because a complete LIC is evaluated, any filter kernel can be chosen. In contrast, image-space advection techniques [LJH03, vW03] are restricted to an exponential kernel, which yields lower image quality than the Gaussian kernel typically used in this implementation (see the discussion of filter quality by Weiskopf [Wei09]).

The third stage (advection and blending) consists of the following components: projection of the original, non-rotated vector field onto image-space; semi-Lagrangian image-space advection of the visualization result from the previous time step; and blending of the advected image with the image from stage two. The projection of the vector field is similar to the projection in stage two. Blending is a simple 2D image operation. Instead of the inexpensive blending operation which results in an exponential filtering, a generic filter kernel can be used here as well. To do this, the same steps are necessary as for the temporal filtering in 2D. In particular, intermediate images of the second stage are stored and a weighting and accumulation process is performed using an arbitrary filter kernel. Similarly to the 2D algorithm, the alternative approach of reducing LIC length in regions of inconsistency can be implemented by modifying stage two; in this case, stage three is not needed.

Semi-Lagrangian image-space advection can cause problems due to inflow at silhouette lines. To avoid inflow of background color, a modified bilinear interpolation is employed within the previous visualization image. This special filter works basically the same way as the standard bilinear filter—except for background texels, which are weighted zero. To decide whether a texel belongs to the background or to the surface geometry, the same mask is used as in stage two. If all four texels lie on the background, a gray-scale value of 0.5 is assumed. Currently, internal edges are neglected, i.e., image information could be transported across such edges. The approach of Laramee et al. [LJH03] could be included to overcome this issue. For the final display, the texture from stage three is modulated by a rendered image of the surface geometry to simultaneously show the vector field texture and the surface shape. This implementation supports the Blinn-Phong model and cool-warm shading [GGSC98] for surface illumination. Bump mapping is also available as an option to emphasize the structure of the vector field texture, mimicking the embossing of flow structures [UIL$^+$04]. Here, the resulting texture from stage three is interpreted as a height field that perturbs the normal vectors.

3.5 Image Compositing of Combined Approaches

Conventional LIC images that show lines along stream lines have the advantage that the direction of the flow is directly perceived in such an image. In contrast, the proposed orthogonal LIC approach has the advantage that the spatial frequency of the moving pattern can be tuned to allow optimal perception of the animated flow. In this section, the computation of conventional tangent-vis within the proposed framework is presented and then image-compositing methods are proposed to combine tangent-vis with ortho-vis. The motivation is to create an approach that benefits from the two different LIC methods and overcomes their respective drawbacks.

To begin, the modifications required to compute intermediate tangent-vis images are discussed. Since tangent-vis and ortho-vis mainly differ in the direction of their field lines, only the changes of the ortho-vis pipelines for 2D (Section 3.3) and 2.5D (Section 3.4) are of interest. The most important difference is that the original vector field, not the rotated vector field, is used: stage two in Figure 3.5, which computes orthogonal LIC, is replaced by a LIC computation that works with the original vector field. Similarly, the 2.5D algorithm avoids the rotation of the vectors before projection to the image plane. Otherwise, the principal processing pipelines are identical for ortho-vis and tangent-vis, including stage one (transport of noise) and stage three (temporal filtering).

The next aspect concerns the actual combination of the orthogonal and tangential representations. These two representations are combined by compositing in image-space using the LIC density (i.e., the gray-scale value that originates from the LIC computation). The LIC density computed for the orthogonal and tangential LIC images is denoted as I_{ortho} and I_{tangent}, respectively. Often, those LIC images are immediately interpreted as gray-scale values for final display. However, this approach takes into account the possibility of some additional color map applied to the LIC images before the final result is produced. The respective color-mapped images are denoted C_{ortho} and C_{tangent}.

Based on this terminology, two different compositing models are proposed. The first one is called the *engraving* model and is formulated by the compositing equation

$$C_{\text{comp}} = \begin{cases} C_{\text{ortho}} & \text{if } I_{\text{ortho}} > I_{\text{tangent}} \\ C_{\text{tangent}} & \text{otherwise} \end{cases}, \tag{3.13}$$

which is applied to each pixel independently. The engraving approach essentially results in a maximum computation: the maximum of the two LIC densities fully determines the final color. For the special case of an identity color map, which implies a gray-scale image, Equation 3.13 is reduced to the maximum operator $I_{\text{comp}} = \max(I_{\text{ortho}}, I_{\text{tangent}})$. This expression motivates the term "engraving": the LIC values can be interpreted as depth values of a height-map; only the deepest engravings survive the compositing (i.e., the subsequent application) of intermediate engravings. Visually, the engraving effect is especially clear in combination with bump mapping, which is demonstrated in Section 3.7.2 and Figure 3.11.

The engraving approach is easily computed, it has a simple interpretation, and it immediately works with height-maps and bump mapping for embossing. However, one possible shortcoming is that engraving only keeps the LIC lines with large values, i.e., dark lines of one representation can never occlude bright lines of the other representation. To overcome this problem, an alternative compositing scheme is proposed, called *interweaving*. The basic idea is that the color and brightness of LIC structures should be chosen independent of their relative depth, leading to interweaving patterns reminiscent of the interweaving of threads in fabrics. Independence is achieved by adding height-map textures H_{ortho} and H_{tangent}, respectively. The height-maps are produced by the same pipeline as the original LIC density textures, but are based on completely independent input noise. The compositing equation of interweaving is

$$C_{\text{comp}} = \begin{cases} C_{\text{ortho}} & \text{if } H_{\text{ortho}} > H_{\text{tangent}} \\ C_{\text{tangent}} & \text{otherwise} \end{cases} .$$

According to the value stored in the height-map texture, either the tangential LIC image or the orthogonal LIC image is drawn. By doing this, LIC images of both types are drawn on top of each other in an alternating fashion, thus creating the desired interweaving effect. The input noise for the height-map computation is always based on filtered white noise in order to densely cover the whole domain by height-field values, without any holes. The height-map texture provides a certain amount of control over the appearance of the final interweaving pattern: by modifying the scale of the white noise the width of the resulting (invisible) depth LIC lines can be controlled, and therefore how often one of the visible interweaving LIC lines “changes depth”.

In contrast to the height-map textures H, the density LIC textures I may use different noise models. For example, moderately sparse noise together with asymmetric filtering is recommended in order to visualize downstream flow direction for tangent-vis according to the idea of Oriented LIC (OLIC) [WGP97]. For visual consistency, the ortho-vis image should also be based on a similar kind of sparse noise in this case. However, there is no natural downstream direction for the orthogonal vector field and, thus, a symmetric filter kernel like the Gaussian function should be applied for ortho-vis.

Different parameters and compositing models for combined LIC are compared in Figure 3.6. An example of interweaving is depicted in Figure 3.6a), where LIC lines of one type are running on top and below LIC lines of the other type in an alternating fashion. The data set is a circular flow. Figure 3.6b) applies

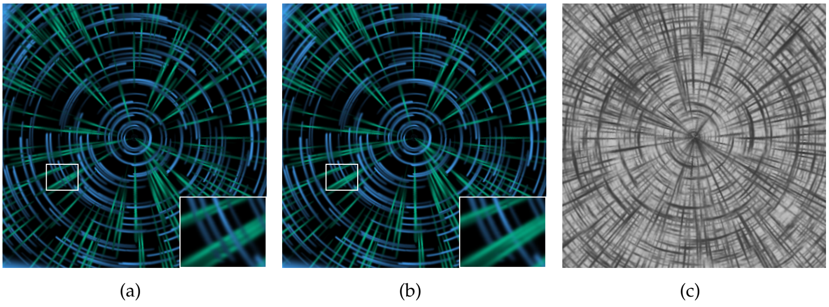

Figure 3.6: A circular vector field visualized with combined LIC: (a) interweaved LIC, (b) engraved LIC, (c) engraved LIC with dense noise.

the engraving model to the same visualization scenario. Zoomed-in views of these two figures demonstrate that engraving and interweaving lead to different occlusion behavior. Similarly to color weaving [UIM$^+$03], appropriate color mapping is recommended for effective engraving and interweaving: the two combined LIC images use two different colors that are highly saturated and approximately isoluminant. Approximate isoluminance is instrumental in avoiding induced perception of shape and other features that could affect the visualization. Isoluminant colors can be conveniently generated by the method of Kindlmann et al. [KRC02]. The colors are chosen to be highly discriminable in order to achieve easy separation of the line structures and good visual continuity [IG97]. In addition, colors are chosen that induce the same depth perception to prevent conflicting depth cues (e.g., red and blue would be bad choices because they induce different perceptual depth).

Figure 3.6c) shows another example of engraving. Here, sparse noise for the density LIC images is replaced by white noise to achieve a denser visual representation. In addition, the isoluminant color table is substituted by a grayscale mapping. Although bump mapping or embossing is not applied, this image provides a subtle depth impression due to perceptually induced shape recognition connected to luminance changes. Gray-scale mapping is recommended only in combination with the engraving model because this compositing scheme guarantees consistency of grayscale values and depth. Furthermore, Figure 3.6c) demonstrates that combined LIC also works with dense noise.

3.6 Implementation

The GPU implementations of the 2D and 2.5D algorithms are based on C++, DirectX 9.0, and HLSL for shader programming. The above algorithms are mapped to vertex and pixel shaders, the data structures are realized by 2D or 3D textures. Shader model 3.0 is essential because loops in pixel shaders are necessary.

For the 2D implementation of ortho-vis with incremental temporal filtering, each operation in the pipeline of Figure 3.5 is mapped to one pixel shader program that works on 2D images represented by 2D textures. Similarly, the analogous pipeline is implemented for tangent-vis, except for the missing rotation of the vector field. For steady vector fields, multiple render passes are not required for any of the stages. The template noise is precomputed on the CPU and low-pass filtered in Fourier space. Data between different stages is transferred as 2D textures (16-bit floating point format) filled by means of the render-to-texture functionality. Semi-Lagrangian advection uses the built-in bilinear interpolation within 16-bit floating point textures.

For the 2.5D implementation with incremental temporal filtering, each stage is essentially mapped to two shaders and two render passes: one shader implements the different variants of projecting the vector field onto the image plane; the subsequent shader is responsible for the actual particle tracing and/or integration. Data between stages is transferred as 2D textures with 32-bit floating point format. Semi-Lagrangian advection is based on a modified version of bilinear interpolation (see Section 3.4) that is explicitly implemented in a pixel shader. Once again, ortho-vis and tangent-vis are based on essentially the same processing pipelines and implementations, except for the difference in the computation of the vector field.

Image compositing for combined LIC is implemented by one image-space operation that takes the intermediate results from ortho-vis and tangent-vis as input and that outputs the composited image. The engraving model works directly on the LIC density values of the intermediate images. For the interweaving model, additional height-map textures have to be generated during the computation of ortho-vis and tangent-vis. Here, it is not necessary to use multiple render passes to create the additional textures. Instead, the additional rendering is performed in parallel by writing to multiple render-targets.

Time-dependent flow requires minor changes in the implementation—the first stage is modified, since the coherent noise is accumulated over past time steps. Here, for each of these time steps, the vector field is found by interpolating between the previous and next time step of the flow. The noise values are accumulated by using multiple render passes. The second and third stage remains unchanged.

The generic temporal filtering process requires a few changes to the aforementioned processing pipeline. First, intermediate, temporally unfiltered images of different times are stored in additional textures. Those images are the result of the second stage in the pipeline of Figure 3.5. Stage three (semi-Lagrangian advection and alpha blending) is disabled. The textures with the unfiltered images are organized as a ring buffer that keeps track of the time tag of the buffer textures: the oldest texture is overwritten with the current result of the second stage. The number of elements in the ring buffer determines the filter size for temporal filtering. The actual filter process reads all unfiltered intermediate images, applies the convolution kernel, accumulates the filtered values, and writes the final result. This is implemented using multiple render passes: the images stored in the ring buffer are weighted and accumulated inside this loop with ping-pong rendering. Therefore, the complexity of the generic temporal filtering algorithm is linear with respect to the filter size, whereas the incremental filtering by recurring alpha blending is independent of the filter size (i.e., the alpha value).

The alternative approach of shortened LIC lines needs an additional texture which is precomputed to store the amount of inconsistency at all texel locations. During the second stage, this texture is accessed to retrieve and apply the scaling factor for the LIC length.

3.7 Results

3.7.1 Performance Results

The following tests were conducted on a Windows PC with Intel Dual Core CPU (1.86 GHz), 2 GB RAM, and NVIDIA GeForce 7900GS GPU with 256 MB of texture memory. Figure 3.7 documents timings (in milliseconds) for rendering a single image of orthogonal vector field visualization. Each plot shows measurements for a 2D data set (Benard convection, depicted in Figure 3.8) and a 2.5D data set (a spherical object with a vector field given on a 3D texture). Figure 3.7 (left) shows the behavior for varying integration length in the

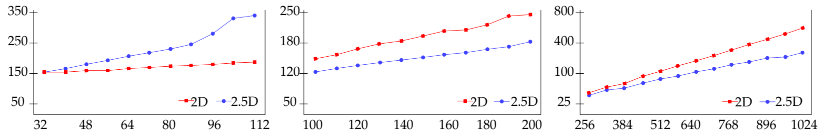

Figure 3.7: Performance results for varying parameters and squared viewports; number of virtual noise textures (left), LIC integration length (middle), resolution (right). All vertical axes show rendering times in ms/frame.

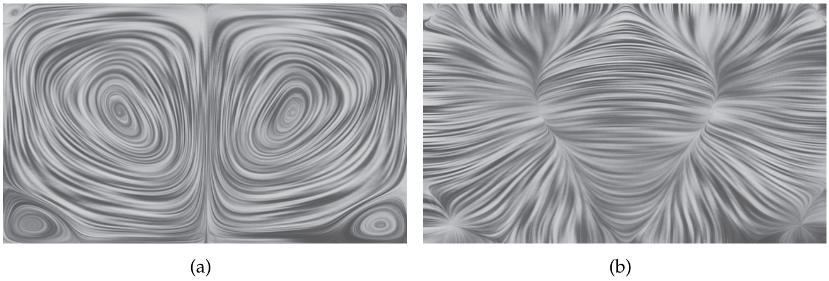

Figure 3.8: 2D vector flow visualized in two ways: (a) conventional tangent LIC, (b) orthogonal vector field visualization.

first stage of Figure 3.5 (noise transport), i.e., different temporal convolution of virtual input noise images. The viewport size is 512^2 and the LIC convolution length is 2×100. The 2D case exhibits an almost linear behavior. The 2.5D algorithm shows an unexpected increase of rendering time for long convolution lengths, which might be explained by an influence of the texture cache.

Figure 3.7 (middle) reports timings for varying LIC integration length (for stage two of Figure 3.5). The integration length is given as the length in one direction, i.e., the total number of integration steps is twice the displayed number. The viewport size is 512^2 and the temporal convolution length is 16. Finally, Figure 3.7 (right) illustrates the influence of the viewport size, for constant temporal convolution length (16 steps) and constant LIC convolution length (2×100 steps). Please note that the y axis has a quadratic scale. As expected, all plots show almost linear behavior for the varying parameters. Therefore, quality (i.e., longer integration or more pixels) can be gradually balanced with rendering speed. In general, typical 2D and 2.5D visualizations render at some 10 frames per second, which facilitates interactive applications.

When interweaved LIC is enabled, the computation time for rendering one frame increases by approximately 60%, regardless of the other parameters. This is an expected result since additional render-targets are used, and therefore the execution time for the pixel shaders of stages one and two increases.

3.7.2 Qualitative Results

First, the orthogonal vector field visualization is compared with the existing method of tangential stream line LIC. In Figure 3.8, convection flow is used to present the two different LIC approaches for the 2D case. Figure 3.8a) shows the conventional way of visualizing the vector field using the standard LIC approach, whereas Figure 3.8b) shows the method of orthogonal vector field

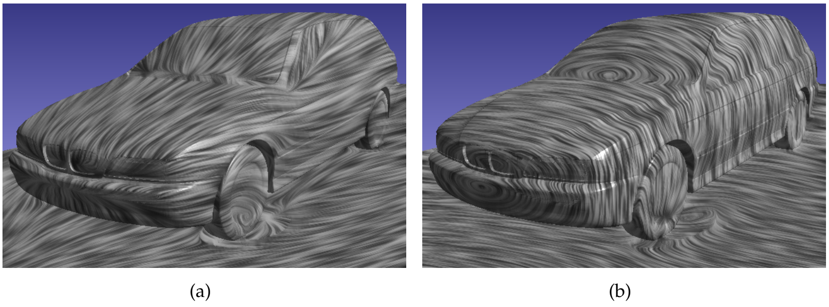

Figure 3.9: Flow visualization on curved surfaces: (a) conventional, tangent LIC, (b) orthogonal vector field visualization.

visualization. Figure 3.9 illustrates tangent-vis and ortho-vis side-by-side for a 2.5D data set from an automotive CFD simulation.

Different compositing models for combined LIC were already discussed in Section 3.5 in Figure 3.6. In addition, with combined LIC several parameters can be changed in order to modify the look of the resulting images as shown in Figure 3.10. Depending on how much detail in the visualization is desired, the density of the sparse noise for the conventional stream line LIC image as well as for the orthogonal LIC image can be adjusted. Furthermore, the integration length for the two different LIC images can be modified.

The orthogonal vector field visualization can also be combined with conventional LIC images in 2.5D: in Figure 3.11a), a uniform vector field is projected onto the surface of a torus. Here, the engraving compositing scheme is applied, which immediately leads to a height-field interpretation. The height-field character is emphasized by bump mapping applied to the composited image (bump-mapping increases the contrast of the lines, making them even better perceivable). In Figure 3.11b), the same technique is used, except for additional color coding of the magnitude of the vector field. The color image is blended with the LIC image, providing an additional cue for the velocity of the vector field.

3.7.3 Temporal Filtering

Figure 3.12 shows the 2D orthogonal vector field visualization of a shear flow. As discussed in Section 3.2.3, a shear flow is a challenging example because of substantial inconsistencies in the transport of orthogonal LIC patterns. In Figure 3.12a), the inexpensive exponential filter is used with an alpha value of 0.075. When alpha blending is being used to perform temporal filtering, a

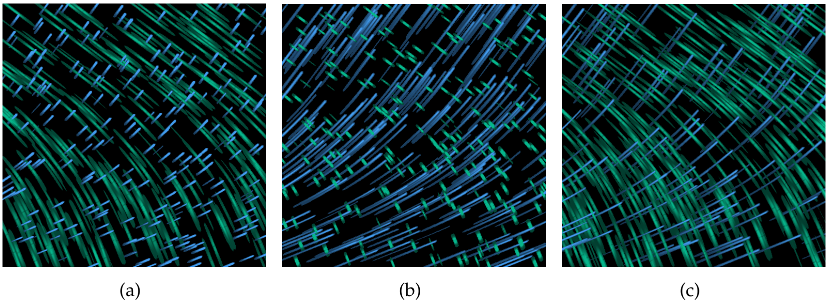

Figure 3.10: Different parameters for interweaved LIC: (a) integration length for conventional LIC lines four times smaller than for orthogonal LIC lines, (b) integration length for orthogonal LIC lines four times smaller than for conventional LIC lines, (c) density of orthogonal LIC lines is three times higher than for conventional LIC lines.

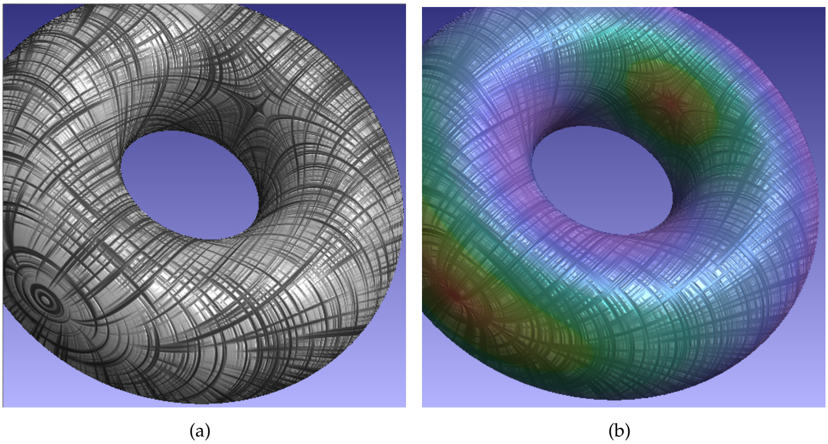

Figure 3.11: An example of combined LIC visualizing the tangential vector flow on the surface of a torus: (a) bump mapping improves the perception of the LIC patterns, (b) velocity magnitude is additionally encoded by using a color map.

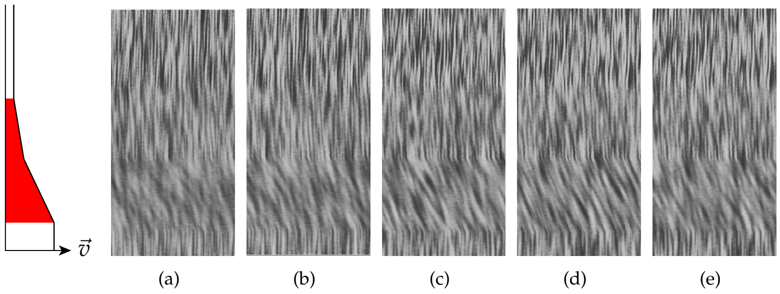

Figure 3.12: Visualization of shear flow with different filters. The graph on the left shows the velocity profile for this example. (a) Recurrent alpha blending with $\alpha = 0.075$, (b) exponential filter with the same falloff rate as in (a), (c) Gaussian filter, (d) sinc() filter, (e) box filter.

smaller alpha value leads to a wider filtering and thus to a larger blurring of the image in regions of inconsistency. This is helpful in perceiving the overall motion of the vector field. From experience, useful alpha values are in the range of 0.03–0.3, depending on the animation speed and structure of the vector field.

In Figure 3.12b), again an exponential filter is used; however, this time the version is used that results from the generic filtering approach with 25 intermediate images. This version shows slightly less pronounced artifacts because the generic filtering pipeline performs a completely Lagrangian particle transport. In contrast, the alpha blending of the incremental filtering process implies the use of semi-Lagrangian advection, which leads to numerical diffusion (see discussion of Weiskopf et al. [WEE03]). Figure 3.12c) is created with a Gaussian filter kernel using the generic filtering pipeline. This kind of filtering reduces irritating patterns in the region of highest velocity because the Gaussian function exhibits a faster falloff in frequency space than the exponential function [Wei09]. These irritating patterns can be observed in Figures 3.12a) and b) as slightly undulating patterns.

Figure 3.12d) shows the result of filtering with the sinc() kernel. In theory, the sinc() filter provides perfect temporal low-pass filtering; however, the filter has to be of infinite length in order to reach this quality. A relatively short filter length of 15 is used in this implementation and the rest is cut off of the infinite filter function. Nevertheless, the results show better visual quality than the exponential filter of length 25. Finally, Figure 3.12e) illustrates box filtering, which serves as a negative example: the region of high velocity is washed out and additional aliasing patterns appear. The reason for the bad quality of

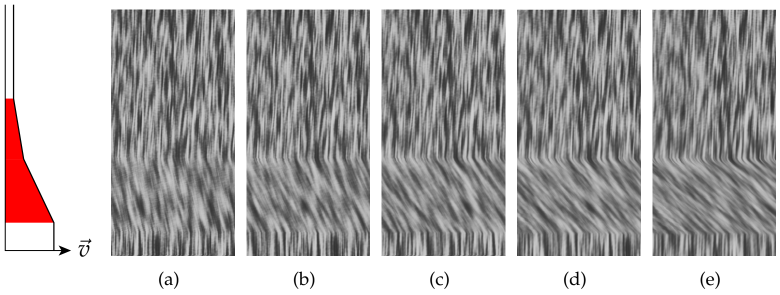

Figure 3.13: Comparison of Gaussian filter kernels of varying length. The graph on the left shows the velocity profile for this example. (a) Length 10, (b) length 20, (c) length 30, (d) length 40, (e) length 50.

the box filter is its high contents of spurious high-frequency contributions in frequency space. In summary, Figure 3.12 demonstrates that the generic filtering pipeline along with appropriate low-pass filters (like Gaussian or sinc()) provides best quality. Nevertheless, exponential filtering—even in the form of incremental alpha blending—leads to acceptable results. Therefore, the incremental filtering process is an acceptable option for fast, interactive visualization.

Figure 3.13 demonstrates the effects of increasing filter length. In this case, the Gaussian filter is chosen and its kernel length is varied from 10 up to 50. As the filter length increases, more images of previous time steps are included in the filtering process. For each rendered frame, the filtering process advects the images of the filter support along the original vector field (see Section 3.2.3 for more details). Therefore, an increase in the slope of the LIC patterns can be seen in regions of increasing velocity. This behavior is a by-product of the proposed temporal filtering technique and can be observed regardless of the choice of filter kernel. This effect might be misleading in static images, since the direction of the flow in the shear area seems to change for increasing filter lengths. However, according to observations, this misinterpretation is less likely in the animated visualization. The advantage of an increased filter length lies in its improved temporal coherence. Based on experience, insufficient temporal coherence results in clearly visible shower-door effects, whereas stronger temporal coherence with a filter length of approximately 30 or greater reduces the shower-door effects almost completely. Additionally, the hypothesis is put forth that strong temporal coherence and reduced shower-door effects facilitate the perception of consistent flow; however, this hypothesis requires further investigation by user studies.

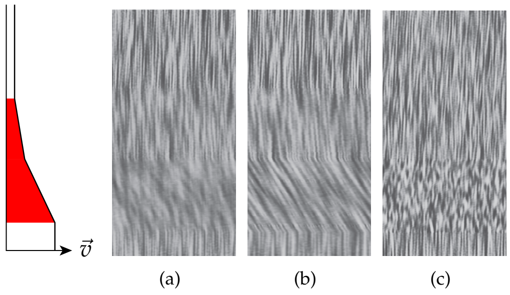

Figure 3.14: Different ways to overcome the problem of shearing flow. The graph on the left shows the velocity profile for this example. (a) Recurrent alpha-blending approach, (b) Gaussian filter, (c) orthogonal LIC lines are shortened in regions of shearing flow.

Yet another way of overcoming the problem with shearing flow is shown in Figure 3.14. For comparison, a) and b) show the results of recurrent alpha blending and a Gaussian filter, respectively, whereas c) was generated without any temporal filtering. Here, the length of orthogonal LIC lines is modified proportionally to the strength of the shear in the visualized vector field. This way, the lines become very short in regions of strong shear to avoid that the LIC lines are "torn apart". The disadvantage of this approach is that it works only with animated visualization—in static images, the underlying vector field is harder to recognize because the line patterns become degenerate.